uc3m | Universidad Carlos III de Madrid

Grado Ingeniería Informática

2016-2017

Trabajo Fin de Grado

Desarrollo de una herramienta web para la resolución de problemas de decisión multicriterio con AHP

María Santana Águila

Tutor

Antonio Berlanga

Salón de Grados

16/10/2017 09:30

Autor: María Santana Águila

Tutor: Antonio Berlanga

Tribunal:

Presidente:

Vocal:

Secretario:

Una vez realizado el acto de defensa y lectura del trabajo fin de grado el día de de , en la Universidad Carlos III de Madrid, Campus de Colmenarejo, acuerda otorgar una calificación de

Presidente Vocal Secretario

Agradecimientos

Después de cuatro años, llega el momento de redactar este trabajo fin de grado con todas las ganas del mundo y tan esperado tanto por muchos como por mí. Han sido unos años de un alto aprendizaje tanto personal como profesional dado que al afrontar una carrera de estas dimensiones debes tener las cosas claras y estar muy segura de ti misma para poder superarla, así como tener constancia de la relevancia que va a tener en tu vida y la necesidad de aprender.

Para empezar, debo rendir mi agradecimiento a muchas personas que me han acompañado a lo largo de mi trayecto por la universidad Carlos III. En primer lugar, quería destacar la Universidad Carlos III, donde decidí estudiar por el prestigio de esa facultad, por la fama y los profesores. Gracias también por brindarme la opción de esas enormes aulas y el material que se nos pone a nuestra disposición, sin el cual quizá hubiera sido casi imposible lograr muchas de las prácticas realizadas, sobre todo las de laboratorio.

En segundo lugar, destacar a los profesores que he tenido el privilegio de disfrutar, son unos grandes profesionales. Me gustaría destacar a algunos de ellos que han hecho de mí aparte de una profesional como todos, una persona capaz de afrontar el mundo laboral como soy ahora. Miguel Ángel Patricio, gran profesor, inmejorable y mejor persona, me ha ayudado en decisiones laborales y me ha aconsejado en el ámbito académico de la mejor forma posible. David Palomar, gran profesor, solo tuve la suerte de cursar una asignatura con él, pero he aprendido como si fuera una cada año, gracias. Por último, pero no por ello el menos importante, Antonio Berlanga, mi tutor del trabajo, que siempre ha estado accesible en cualquier momento y confió en mí para desarrollar un aplicación de alto impacto actualmente usado el algoritmo en muchas empresas.

También quería mencionar y resaltar a mis amigos que me han acompañado en la trayectoria y toda mi familia que ha aguantado estos 4 años a mi lado dándome los mejores consejos y levantándome cuando me caía para seguir adelante.

Quiero dedicar la carrera a toda mi familia que es lo mejor de mi vida, y en especial a mi abuela que se que está orgullosa de mi.

¡Muchas gracias a todos!

Resumen

El proyecto desarrollado a lo largo de este documento corresponde a la implementación de un algoritmo de resolución de problemas de decisión multicriterio.

Hoy en día en cualquier empresa se tienen que tomar decisiones, al igual que cada persona las toma de forma cotidiana, por lo que voy a desarrollar una aplicación que nos ayude a tomar las decisiones de la mejor forma posible.

El principal aliciente para la implementación de esta herramienta web radica en la necesidad del día a día de toma de decisiones. La facilidad que nos ofrece a través de la app es inmensa que ya iremos viendo a lo largo del trabajo pero principalmente se basa en meter tus dudas y según los criterios que tú valores podrás encontrar la mejor solución.

Voy a poner un ejemplo para que se entienda de una manera mucho más sencilla: en cualquier empresa, el departamento de recursos humanos se encuentra a diario con un problema de decidir si es mejor un candidato u otro para un puesto. Con esta herramienta bastaría con introducir los nombres de los candidatos y las características que debe cumplir para acceder a ese puesto, rellenando cada uno de los campos, saldría el perfil que más se adapta. Si el día de mañana se busca otro perfil seria cambiar las características que deben cumplir y su importancia y saldría un nuevo perfil.

Por poner un ejemplo real, este algoritmo lo están usando en muchas empresas, en concreto el Ayuntamiento de Madrid pide unos excel para saber que barrio es el más vulnerable o el que más población tiene, con esta herramienta bastaría con introducir los datos y tendríamos la solución, sin necesidad de crear una hoja excel por cada estudio.

Abstract

The project that has been developed through this document is about the implementation of a solve problem algorithm for multicriteria decision.

Nowadays every company has to make decisions just as everybody makes their own choices everyday, so I am going to develop an app to help us making these decisions as good as possible.

The main incentive for the implementation of this web tool consists in the necesity of everyday choices. The benefits of this app is huge as we will realize through this project but mainly it is about introducting different criteria and with it you will get the best solution.

I am going to explain it with an example: in the companies, the Human Resources department deals everyday with which is the person that better fits for the position. With this app you just simply have to introduce the names of the candidates and the caractheristics requested for the position ,filling all the fields the result that the app will give it to you should be the best option. For future positions , you just have to change the characteristics requested and their importance and you will have a new profile.

A real example: this algorithm is being used in many companies. In particular, the Ayuntamiento de Madrid requests excels documents to know which neighbourhood is more vulnerable or the most populated. With this app you just have to fill the fields and you will have the solution, no need of creating an excel document for every study.

Índice

Índice de tablas

Índice de ilustraciones

1. Introducción

A día de hoy y siempre como ya he comentado en el resumen anterior, las decisiones están de nuestra mano desde que nos despertamos por la mañana. Dado el avance de la tecnología y en el momento en el que nos encontramos resulta extraño que no se haya creado una herramienta web para la resolución de este tipo de problemas con inteligencia artificial empleando el algoritmo de decisión multicriterio.

Existirán más de una aplicación web y móvil para la resolución de este tipo de problemas, pero mi propuesta consiste en facilitar y fomentar la práctica de la inteligencia artificial y sus algoritmos para la resolución de dicho problema.

1.1 Contexto Actual

Las ventajas que nos proporciona la tecnología, nos ha servido de gran utilidad en nuestra vida. Dentro del mundo comercial la tecnología juega un papel influyente, está claro que nos encontramos en el "Mundo 2.0" ante una nueva fase de desarrollo humano junto a la tecnología y con una fuerte influencia y tendencia a la inteligencia artificial.

Tal es la velocidad con la que evolucionamos hoy en día que no nos da tiempo a pararnos a pensar como está cambiando nuestros días. Hace unos años era impensable el ir en un coche compartiendo viaje con otra persona que no conocieras (Blablacar) o poder pagar con el móvil sin necesidad de introducir un código secreto (Twypcash).

Dentro de unos años, no muy lejos, empezaremos a no tener tarjetas de crédito para pagar, todos los procesos estarán cada vez más automatizados y se hará un mayor uso de las máquinas y de la tecnología que de las personas.

Por otra parte, centrándonos más en el ámbito que voy a tratar en este documento, las decisiones se toman y son necesarias tanto en empresas pequeñas como grandes así como en cada uno de nosotros. Hablo como si se tratase de decisiones siempre pero también englobo en este campo todos y cada uno de los informes que se quiera estudiar sobre una característica concreta de distintos casos.

Vamos a estudiar lo que el mercado actual en este sector pone a nuestra disposición para valorar en que me voy a diferenciar al implementar esta y no otra aplicación.

Contamos con una aplicación desarrollada con el fin de dar facilidad a los senderistas de crear rutas en función de su motivación, dificultad y capacidad física. La aplicación ejecuta un sistema recomendador basado en el análisis de redes y en el algoritmo de búsqueda A* para calcular las rutas que cumplan con los criterios del usuario.

Otro caso de uso de dicho algoritmo es un proyecto desarrollado en la universidad. Las listas de espera son un problema para la mayor parte de los países que cuentan con un Sistema Nacional de Salud. Este desarrollo propone analizar el problema de las listas de espera desde una perspectiva de Decisión Multicriterio. Tras un análisis de las diferentes metodologías existentes, se decidió elaborar un modelo de decisión basado en el método AHP (Analytic Hierarchy Process) para la gestión de listas de espera y ahora mismo se está aplicando en los Hospitales de la Comunidad de Madrid.

Un caso de uso más es la aplicación a un problema de generación de energía eléctrica que fue desarrollado en 2013 para evitar y disminuir la complejidad y argumentos a la hora de toma de decisiones.

Este proyecto surge con la idea de actualizar el ámbito de decisiones en el mundo tecnológico en el que nos encontramos. Hoy en día todos o casi todos disponemos de un teléfono móvil, tablet u ordenador a nuestro alcance para poder acceder a esta app.

1.2 Descripción del proyecto

El proyecto consiste en la implementación de una herramienta web basada en el algoritmo de decisión multicriterio, hoy en día este algoritmo, como ya he mencionado en el contexto actual, se está usando en muchas empresas o para determinadas aplicaciones pero, ¿por qué no crear una aplicación genérica que nos sirva para cualquier casuística que se nos presente?

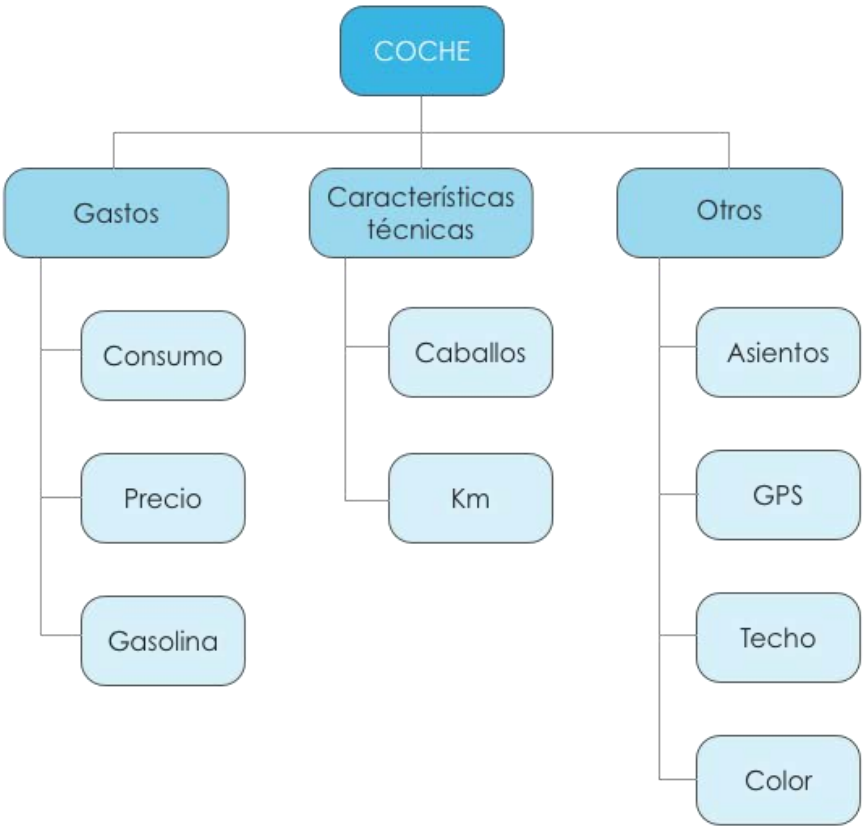
La toma de decisiones requiere niveles de argumentación en función de la información que se tiene disponible y se ha investigado para esa decisión, así como de las consecuencias que conllevará una vez tomada.

Escoger una decisión en lugar de otra es el resultado de un largo de proceso de investigación en el cual es importante conocer las preferencias de cada uno. A parte de muchas características que influyen en la elección, juega un papel muy importante el estado anímico de cada uno de nosotros, si te levantas un día con el pie izquierda puede que tomes una decisión diferente a un día de fiesta y en caliente sin pensar bien o los argumentos. Este papel es difícil de controlar pero al utilizar el algoritmo multicriterio podríamos meterlo como un factor más e influirá en nuestra elección en función de la relevancia que le asignemos.

La decisión multicriterio MCDM (Multiple Criteria Decision Making) es la teoría que estudia y analiza los problemas de decisión que involucran dos o más criterios de evaluación. La MCDM radica en enmarcar con la mayor precisión posible, problemas reales de decisión que involucran diferentes criterios, en los cuales es muy difícil encontrar la decisión correcta entonces debe ser el decisor, aplicando diferentes técnicas, quien aplique un peso a cada uno de los factores que contempla y de esta forma elija la mejor solución dentro del conjunto de soluciones factibles.

El fundamento de AHP que he empleado en la aplicación se basa en atomizar los problemas e ir agregando las soluciones de los mismos. Es decir, el primer paso para la aplicación de este método es estructurar jerárquicamente el problema en diferentes nodos interconectados.

Adjunto un árbol de ejemplo de los niveles que podemos establecer y a continuación una breve descripción de cómo se leería ese árbol en función del algoritmo y el fundamento AHP explicado en el párrafo anterior.



Como podemos observar en el gráfico anterior a modo de prototipo de árbol que crearíamos en la aplicación, el nivel superior se trata del propósito del problema, "la decisión final". Los niveles intermedios, es decir, los nodos que no son hoja del árbol, corresponden a las diferentes agrupaciones que hacemos de nuestro factores según nos vamos acercando al objetivo nos vamos abstrayendo unos niveles. Por último, los niveles hojas son las características que yo tengo de cada uno de mis candidatos a la elección, los factores que voy a valorar y me van a decantar por una decisión y no por otra.

La aplicación de AHP sobre el árbol implica:

- Realizar comparaciones por pares entre los nodos de cada nivel jerárquico, en base a la relevancia que el usuario considere que presentan para el nodo del nivel superior de la jerarquía al que están ligados. Los resultados de estas comparaciones se recogen en forma de matrices de comparación por pares.

- Obtener cada uno de los vectores de prioridad correspondientes a cada una de las matrices de comparación por pares.

- Con las matrices dadas en el paso anterior se calcula la contribución de cada alternativa al objetivo del problema, mediante una agregación multiplicativa entre los niveles jerárquicos y en función de estos resultados, se ordenan los candidatos a decisiones y se selecciona el que más encaja dentro de lo establecido.

- Como último dato debo añadir que una vez dado el resultado que más se adecue según los criterios que estamos valorando, es posible realizar un análisis de sensibilidad del resultado alcanzado, visualizando y analizando cada matriz. Por lo que se puede hacer el cambio que se desee para obtener otro resultado o siguiendo otros criterios si ahora han cambiado por diversos motivos.

Una vez expuesto el algoritmo por el que orbita este proyecto, procedo a entrar en la parte técnica de su desarrollo en grandes rasgos además de en primer lugar mencionar el nombre elegido para la aplicación y el porqué de mi elección.

La aplicación se denominará “Master Search”, porque se trata de un algoritmo de búsqueda de la mejor solución según una serie de criterios que cada usuario establecerá. Además de eso, como anécdota, cuando pensé el nombre que más se adecuase y en una o dos palabras se definiese lo que hace la herramienta, al salirme estas dos coinciden con mis iníciales lo que me ayudo a declinarme por esta opción finalmente.

En la fase de investigación y análisis del contexto actual y las tecnologías punteras a día de hoy hay muchos frentes y muchas decisiones que tomar.

- ❖ Elijo Angular JS como lenguaje de desarrollo de esta herramienta porque:

- Está creciendo exponencialmente y sacando muchas versiones con pruebas ya pasadas.
- Angular JS y no angular 2 porque ha dado más tiempo a probar y no presenta fallos, angular 2 es una versión beta que aun cojea de muchas patas como para yo poder implementarla en un proyecto de este alcance.
- Muchas empresas de este sector lo utilizan como lenguaje de programación hoy en día en los departamentos Front.
- No la he cursado en la carrera y es un lenguaje puntero y necesario a la hora de dedicarte al desarrollo de aplicaciones en el mundo laboral.
- Dispone de librerías de diseño como bootstrap, angular JS material que me permite desarrollar con mayor certidumbre y velocidad mi aplicación.

❖ Utilizar Visual Studio Code como herramienta de desarrollo porque:
- Da soporte en todos los sistemas operativos.
- Herramienta clara y simple visualmente y conceptualmente.
- Fácil de desplegar en el servidor la aplicación creada usando nodejs (bower y npm).
- Puedes contar con la consola para añadir dependencias e instalar librerías con bower.
- Arranca el servidor con la instrucción “gulp serve”.

❖ Librería Bootstrap para Angular JS porque:
- Sencilla y concreta visualmente.
- Basada en componentes atómicos.
- Posee mucha y muy buena documentación.
- Cobertura de más de un 80% en test de cada componente.

A lo largo de este documento iremos viendo en más detalle cada una de las pantallas implementadas, el diseño y la explicación, así como muchas otras cosas que se citan en el índice superior.

Para terminar, a modo resumen de este apartado, se basa en una herramienta web que nos permite dibujar el árbol de decisión, gracias a la implementación, con todas las características citadas, del algoritmo según los pesos establecidos se generará una página con las matrices que nos muestra la solución y podemos ajustarla o ver cuál ha sido el factor determinante por ejemplo.

1.3 Objetivos

En este apartado voy a dividir los objetivos en profesionales y personales para poder ver de una manera más diferenciada lo que me aportan y lo que quiero lograr en los dos ámbitos que son muy importantes ambos. Uno para conseguir el alcance del proyecto y los otros de cara al mundo laboral.

1.3.1 Objetivos profesionales

Una vez conocido el contexto que influye en la aplicación a desarrollar, a continuación se revelan los objetivos a alcanzar a lo largo de la realización del proyecto. Estos objetivos me ayudarán a sacar al final del documento mejores conclusiones y ajustar el grado de cercanía entre el trabajo realizado y el inicialmente esperado.

- Investigar algoritmo multicriterio.
- Crear una aplicación web que implemente el algoritmo multicriterio.
- Realizar un prototipo que cumpla todos los requisitos del cliente.
- Dotar a la aplicación de suficiente diseño como para que sea muy sencilla e intuitiva para el usuario (“UX”: User Expirence).
- Presentar la maquetación al cliente.
- Describir el proceso de desarrollo.
- Implementar la aplicación con angularjs.
- Solventar el problema cotidiano de decisiones en empresa o personales.

- Crear una pantalla de visualización con los resultados una vez analizados los candidatos.
- Servir como buena partida para futuros proyectos.
- Crear un manual de instrucciones/documento en este archivo que cualquier persona que quiera utilizar la aplicación se vea capaz.

1.3.2 Objetivos personales

A parte de los objetivos como proyecto de este trabajo que he citado en el apartado anterior, siempre se deben tener objetivos para crecer como persona y formarte. En mi caso, estos son mis objetivos al comienzo del trabajo:

- Aprender el lenguaje Angular JS.
- Desarrollar una aplicación desde cero sola.
- Aprender a levantar la aplicación en el ordenador.
- Aprender el manejo de Nodejs.
- Aprender el uso de la tecnología npm.
- Aprender el uso y aplicación de bower.
- Aprender a usar librerías de diseño de componentes dadas de un lenguaje.
- Lograr una autonomía y soltura con la tecnología Ajax.
- Lograr una autonomía y soltura con la tecnología JavaScript.
- Investigar y estudiar el algoritmo de decisión multicriterio.
- Investigar y estudiar el fundamento AHP para aplicarlo.
- Investigar sobre las aplicaciones de la competencia.

1.3 Fases del desarrollo

A la hora de desarrollar un proyecto de estas características he seguido las típicas fases en el desarrollo de un proyecto software. Se comenzó con una primera fase fuera del desarrollo software de investigación, estudio y análisis del entorno actual, competencias, y desarrollo del algoritmo multicriterio.

Todo el proyecto tuvo una duración de 6 meses y medio o lo que es lo mismo 141 días laborables. A continuación expongo de manera detallada cada una de las etapas de mi proyecto y la duración de cada una de ellas.

- **Fase de análisis de la competencia**: fue la primera fase para hacerme una idea de la competencia que había en el mercado y a qué me enfrentaba. Tras ver lo que había no encontré una aplicación web de esas características, solo existían, como he descrito en el apartado de "contexto actual", aplicaciones que utilizaban este algoritmo pero para cosas determinadas, por ejemplo para decisiones de rutas de bicicletas pero son para un contexto único. Esto me llevo 3 días de 7 horas, es decir, 21 horas de estricto análisis de competencia.

- **Fase de investigación del problema**: Fue la fase atómicamente de mayor duración en el proyecto debido a la gran cantidad de nuevas tecnologías y muchos conceptos que anteriormente no conocía y tuve que aprender. Tuvo una duración total de un mes, es decir, 30 días naturales con un total de 7 horas al día, sumarían un total de horas de 210. Aquí incluyo el análisis del algoritmo de tal forma de conseguir entender la lógica para llegar a encontrar la mejor tecnología que solvente todos los problemas que me pueda encontrar de una forma sencilla pero completa. Tras esta fase decidí que la tecnología que iba a usar era angularjs que no había estudiado en la carrera pero está muy demandado en el mercado y podía servirme de cara a un futuro. Tras decidir esto los últimos días tuve que familiarizarme y hacer cursos por internet de la herramienta para entender la sintaxis en el desarrollo, lo cual me resultó una gran dificultad hasta conseguir entender la lógica y conexión entre componentes y directivas.

- **Fase de análisis**: se extraen los requisitos del producto de software a desarrollar. En esta etapa es crucial la experiencia y habilidad que se nos ha proporcionado en la universidad para reconocer los requisitos del cliente de forma que se eliminen los incompletos, ambiguos o contradictorios. Habitualmente, el cliente te proporciona unos requisitos muy abstractos y ambiguos a lo que yo, como ingeniera tengo que concretar y negociar con el cliente de modo que cumpla sus necesidades para ayudarle a que obtenga una visión completa de los requerimientos. El objetivo principal de esta fase es la comunicación, una etapa muy intensa en la que hay que eliminar la máxima ambigüedad con él. Tuvo una duración de 49 horas repartidas en una semana (7 horas / 7 días). Durante esta fase obtuve los documentos completos de casos de uso y requisitos de software.

- **Fase de diseño y arquitectura**: determina como funcionará de forma general sin entrar en detalles la aplicación. Consiste en el diseño de un prototipo de la interfaz que se ajuste con los requisitos de la fase anterior y sea usable para el usuario según los criterios de "user experience". Se suelen hacer diagramas previos para saber las interacciones entre las entidades y su secuenciado. Además se definen la arquitectura del sistema, tanto física como lógica de la herramienta. Ocupó una duración de 20 días de una media de 6 horas, es decir, 120 horas en total.

- **Fase de desarrollo e implementación**: básicamente, se resumen en la traducción de la fase anterior, diseño, en código. Es la parte más obvia de un proyecto de software que todo el mundo cuenta con ella y es la primera que nombra, dado que es la fase de la que se van sacando resultados "tangibles". No tiene porque ser la más larga pero sí que es la fase en la que te das cuenta de los fallos cometidos en las anteriores, ya que un fallo en un requisito o diseño puede hacerte retroceder y volver a especificar y retrasar días en esta fase de implementación. Duró 50 días, 6 horas diarias, 300 horas totales.

- **Fase de pruebas**: el principal objetivo de esta fase es asegurarte que la fase de análisis se cumple, realizando una prueba por cada requisito y comprobando que se cumple cada funcionalidad negociada. En ella se plantean unas pruebas recogidas en el plan de pruebas, y se llevan a cabo.

Llevó 15 días, 5 de especificación de cada prueba y 10 de implementación, con un total de 75 horas.

- **Fase de documentación**: esta fase se realiza de forma paralela a todas las anteriores y es en las que se realiza este documento, es decir, como objetivo tiene conseguir una memoria del proyecto completa que dé respuesta a cualquier duda que se plantee alguien que quiera saber del proyecto. Tuvo dos partes; la primera, que se desarrollaba a la par como he comentado para no olvidar lo que iba haciendo que era cada día de los anteriores de cada fase mencionada más 2 horas de documentación, y la segunda, fue al terminar la fase de pruebas en la que se da la aplicación como finalizada para rellenar los apartadas que se deben realizar después como conclusiones, horas de la fase, u otros de estas características. Tuvo una duración en total de 141 días que duró el proyecto repartidas en las dos fases descritas.

1.4 Medios empleados

A lo largo del proyecto para poder llevarlo a cabo he tenido que utilizar medios tantos a nivel de software como hardware que me cubran todas las necesidades de las fases del apartado anterior.

1.4.1 Medios hardwares

He clasificado los diferentes medios de hardware que he utilizado en tres subcategorías:

1. **Hardware para desarrollo de proyecto:**

    a. Un MacBook Pro para el desarrollo del proyecto y el prototipo en Sketch ya que es una aplicación que solo se da soporte en Apple.

b. Una Microsoft Surface para desarrollar a la par la documentación que el código y buscar información.

2. **Hardware para copias de seguridad:**
   a. Un disco duro para hacer backup del código y documento.
   b. Drive para tener la memoria en la nube y que no desaparezca por una pérdida o rotura de ordenador.

3. **Hardware para documentación:** una impresora para imprimir este documento de memoria y poder tenerlo en papel para entregarlo el día citada en la defensa.

1.4.2 Medios software

- **Entornos de desarrollo**
  - ***Visual Studio Code:*** utilizo este entorno porque me permite abrir el proyecto, editar código y automáticamente, una vez se ha inicializado el servidor, se va actualizando solo en el navegador.
  - ***Sketch:*** herramienta para el desarrollo de prototipos web a la hora de pasar a interacción y hacerme una idea más concreta del diseño de la aplicación y los componentes que voy a necesitar.

- **Servidores**
  - ***GulpJS:*** es un automatizador de tareas escrito en Java Script y que corre bajo node.js que sigue la misma filosofía que grunt. Mejora en cuanto a facilidad de programación y rapidez a la hora de ejecutar las tareas.

- **Gestores:**
  - ***Bower:*** gestor de dependencias para el desarrollo de aplicaciones web fronted que me facilita la tarea de instalación y mantenimiento al día las siguientes librerías que he instalado.

- **Librerías**

- ***NodeJS:*** librería y entorno de ejecución de E/S dirigida por eventos y por lo tanto asíncrona que se ejecuta sobre el intérprete de JavaScript creado por Google V8. Me ayuda de una forma sencilla y rápida a ejecutar JavaScript en el servidor.
- ***Karma:*** librería que se encarga de ejecutar los test unitarios de cada una de las funcionalidades de la aplicación.
- ***Angular:*** framework MVC de JavaScript para poder desarrollar mi aplicación SPA (Single-Page Applications).
- ***Angular-ui-tree:*** es un componente de angular-ui, que me permite desarrollar el árbol de decisión de una forma mucho más sencilla.
- ***Angular material:*** librería que me proporciona css para los componentes de la aplicación, directamente puedo coger un componente y ya tengo instalada su clase y se mostrará con el diseño que venga. Si quiero luego puedo modificar y personalizar colores o tamaños.

- **Software de documentación**
  - ***Microsoft Office 2017:*** para realizar la memoria del proyecto.
  - ***StartUML:*** para crear los diagramas UML.
  - ***Microsoft Office PowerPoint 2017:*** para realizar la presentación con la que voy a exponer mi trabajo el día del tribunal.

1.5 Estructura de la memoria

En este apartado del documento voy a mencionar cada capítulo del proyecto con una breve descripción de cada uno de ellos, con el fin de facilitar la búsqueda y la lectura de algo concreto.

1. INTRODUCCIÓN

Recoge una visión general sobre el proyecto, que te permite de una forma rápida hacerte una idea del concepto que se va a desarrollar y para que puedes utilizar la aplicación. Dentro de este apartado, primero te sitúa el contexto actual de este sector, luego una descripción del proyecto seguido de los objetivos, las fases de desarrollo y los medios que he utilizado para el desarrollo.

2. ESTADO DEL ARTE

Te permite ver siempre dentro del mercado en el que se encuentra el proyecto, un análisis de las aplicaciones similares y posibles tecnologías y por último se explican las principales tecnologías que he elegido para la resolución del problema.

3. ENTORNO DE DESARROLLO

Sección destinada a la explicación y registro del software del proyecto así como del hardware que he necesitado para el estudio, desarrollo e implementación del proyecto. Además se incluye una guía de como arranqué el proyecto, la configuración inicial, lo necesario para lanzar la aplicación web y cómo ir haciéndolo una vez tengo todo configurado.

4. ÁNALISIS DEL SISTEMA

En este capítulo se hace un estudio de la propuesta de proyecto por parte del cliente. Se analizan los stakeholders y se procede a desarrollar un documento de casos de uso con el diagrama y los actores identificados para posteriormente sacar los requisitos de software funcionales y no funcionales y relacionarlos con los casos de uso anteriores.

5. DISEÑO

Dentro de este apartado hay dos grandes secciones: la primera se trata de la arquitectura del sistema en la que detallo cada uno de los componentes de la aplicación junto con diagramas explicativos para ilustrar mejor la decisión elegida. La segunda gran sección contiene el diseño de la interfaz web, en la que se adjuntan describen los principios de diseño seguidos y los prototipos realizados con Sketch y una explicación de cada uno de los diseños.

6. PRUEBAS

Apartado destina a la redacción de cada una de las pruebas realizadas y ejecutadas con Karma a lo largo del proyecto. Se recogen en una tabla plantilla cada caso de uso y una fila por cada prueba unitaria del caso de uso repetida tantas veces hasta que su estado sea completado.

7. PLANIFICACIÓN Y PRESUPUESTO

Compuesta por la parte de planificación en la que se describe cada una de las fases del proyecto con sus respectivas horas y el total en días y meses que ha tenido de duración el proyecto. En esta primera parte se adjunta el diagrama de Gantt que muestra de forma gráfica el transcurso del desarrollo de la app web. La otra parte del capítulo se trata del presupuesto para llevar a cabo toda esta aplicación web.

8. CONCLUSIONES Y MEJORAS

Último capítulo de la memoria, en el que se exponen las posibles mejores de cara a un trabajo futuro o en función de cada uno de los clientes que quieran contar con la aplicación para su trabajo diario. También se citan las conclusiones sacadas revisando si se han cumplido las expectativas del cliente, las tecnologías aprendidas o datos a destacar.

Glosario de términos

Anexo del documento que se recogen los términos más técnicos o específicos de esta materia para que cuando lo lea cualquier persona pueda recurrir a este glosario y saber que significa ese término que busca.

Referencias

Conjunto de referencias utilizadas para el desarrollo y documentación del proyecto final en el que se recoge por si se necesita ampliar información sobre un tema en concreto citado en este documento.

2. Estado del arte

2.1 Introducción

Como he explicado en el capitulo anterior en el apartado de descripción del proyecto, consiste en una aplicación web que nos permita introducir nuestras posibles candidaturas de elección y en función de los factores que considere necesarios, obtendremos lo que para nosotros es lo mejor.

Para el uso de una tecnología es interesante conocer primero la historia y el futuro de la misma para poder optimizar el desarrollo y no caer en errores que se han cometido antes. Esta sección pretende explicar el estado de la tecnología de aplicaciones web que se pueden encontrar en el mercado.

Una vez hecho esto se expondrá la solución al problema planteado en el proyecto, indicando las tecnologías que se van a utilizar y explicándolas de una forma detallada.

2.2 Estado del arte

Desde que existen los seres humanos, existen las decisiones. Cada minuto de nuestras vidas está definido por las decisiones que tomamos consciente o

inconscientemente. Estas pueden cambiar nuestra carrera, nuestros amigos, nuestros gustos, nuestra vida completa o parcialmente. Incluso al no elegir, estamos decidiendo.

Por desgracia no existe un manual al que puedas acudir a buscar cual sería la mejor decisión, entonces todos nos preguntamos en cada decisión importante, ¿habré elegido la mejor decisión?

Tomar buenas decisiones es uno de los aprendizajes más difíciles que cualquier persona puede tener a lo largo de su vida. Esto se debe a que cualquier situación que se te presente puede tener una decisión que tendrá una consecuencia en el futuro ya sea para bien o para mal.

Por estos motivos que estoy mencionando es vital tener una metodología establecida que nos ayude y nos permita elegir de la mejor manera posible. También es importante recalcar que no hay decisiones perfectas sino con diferentes consecuencias.

Hace unos años, en los inicios de las personas, la toma de decisiones pequeñas a penas se trataban como decisiones sino como el pan de cada día. Y lo único que se consideraba decisión era aquello que tenía como consecuencia un cambio brusco en tu vida.

Vamos a dividir este capítulo en varias secciones para que veamos los ámbitos a gran nivel que afectan decisiones y luego una vez estudiada la evolución veremos cómo resolver y que ventajas nos ofrece en nuestra vida.

2.2.1 En lo profesional

En la vida profesional existen muchos sectores a los que puedes elegir dedicarte pero en todos aunque parezca mentira vas a tener en tu cargo una serie de responsabilidades, lo que te va a llevar a tomar decisiones que afecten en el futuro de tu vida laboral.

Para verlo todo más claro, la mejor forma es poner ejemplos reales de algunos de los sectores más comunes:

- **Departamento de RRHH:** una de las tareas que tienen a su cargo estos trabajadores es la de buscar candidatos para cubrir un puesto vacante o nuevo que se ha generado en la empresa. Para ello en primer lugar hacen una recopilación de posibles candidatos que cumplan los requisitos y posteriormente procederán a hacerles una entrevista. Cuando han hecho este proceso con todos los candidatos al final tienen que decidir quién es el elegido en función de las condiciones de la vacante.

- **Sector sanitario:** los médicos tienen gran responsabilidad y posiblemente será de los empleos con más decisiones diarias. En una consulta el doctor/a cita, por ejemplo, a 40 pacientes diarios. A cada paciente en primer lugar le tiene que escuchar para que seguidamente pueda redactar un informe de sus síntomas y una vez metido todos los síntomas, tendrá que decidir que medicamentos darle, o si ingresarle o remitirlo a otro especialista. Vemos una vez más como se van a enfrentar a más de una decisión.

- **Sector de la construcción:** este caso es muy parecido al anterior pero con otro contexto de fondo, los arquitectos se enfrentan a diferentes proyectos dentro de una empresa, en los que hay muchas elecciones. Un proyecto de una casa tendrán que ver la orientación que se va a construir, el terreno, los materiales, la distribución, … cosas que luego implicarán la compra o no de diferentes grupos de personas, no es lo mismo un chalet de 5 dormitorios que será asequible por lo general por gente de 40 años de edad o si es un piso de 1 o 2 dormitorios y materiales más baratos que podrán adquirirla jóvenes que aun no tengan tanto ahorrado (siempre hablando de una línea general, aunque como en todo, hay excepciones).

- **Ayuntamiento:** dentro de este campo hay muchas cosas que se pueden tocar, entre ellas por ser un poco tangibles sería los estudios de los distintos distritos de Madrid. Si se quiere estudiar cual es el municipio con más robos en la calle, estudiarían diferentes factores y verían cual según esas características es el barrio en el que más se roba. Se podría sacar una lista ordenada de los municipios según las características que hayamos reunido.

- **Sector tecnológico:** la tecnología da para mucho pero voy a ir a un campo un poco más cerrado. En la informática, el departamento de arquitectura

de cualquier empresa del sector, tiene que decidir la base de cómo se va a organizar el código el proceso que se va a seguir, el espacio que disponemos y cómo se va a desarrollo. Así como lo hacen los de arquitectura, entraría en el lado de backend de qué tecnología elegir, de fronted como mostrarlo.

2.2.2 En lo personal

Tal y como he hecho en la sección anterior voy a citar situaciones comunes que cada persona tenemos en la vida personal.

- **Estudios:** en este periodo de la vida de cada uno que dura obligatoriamente 13 años pero puede llegar a durar 19 o más, es un tiempo que está lleno d3e decisiones que tomamos cada uno con ayuda de nuestra familia y amigos. En 3º ESO debes decidir si el año siguiente te vas a especializar en ciencias o letras, luego otro ejemplo sería si continuar con bachillerato o hacer un módulo de grado medio para incorporarte a trabajar. Si has pasado bachillerato pasas a una etapa que a todos nos parece que va a marcar nuestras vidas, como es la elección de la universidad que quieres ir y qué quieres estudiar, por lo que son muchos factores los que valoras y dependiendo de la persona cada uno tendrá un peso diferente.

- **Amigos:** ¿Qué clase de amigos elegirás? Para algunas personas es una labor sencilla la de hacer amigos, pero para otros es una auténtica pesadilla. Este tema se da sobretodo en adolescencia porque tienen una necesidad muy grande de ser aceptados y eso a veces les lleva por el mal camino. Deben aprender a soportar la presión de compañeros y no dejarse llevar por lo que se lleva en el momento sino pensar las consecuencias que puede llevar como decisión.

- **Familia:** la calidad que cada uno tiene con su familia es una elección aunque parezca mentira, y se trata de una de las decisiones más importantes de la vida. Debes saber cómo construir esa relación, y como poder mantenerla. En

este tipo de decisiones personales influye mucho el entorno pero también la personalidad de cada uno y lo que quiera tener en su vida, "sus prioridades".

- **Casarse:** importante elección como todas las personales que he ido citando en esta sección, en este caso estas eligiendo a una persona para compartir tu vida, por lo que te lanzas con alguien que nunca habrás terminado de conocer porque todas las personas en la vida pueden sorprenderte. Por lo que va a depender de factores personales con esa persona y familia pero también dependerá a lo mejor de cercanía, dinero, religión, muchos otros factores que habría que estudiar antes de dar el paso.

2.3 Evolución de la metodología

Como en todo, los métodos de hacer algo evolucionan de la mano de los años, vamos a hacer un pequeño y breve recorrido a lo largo de la historia de cómo se tomaban las decisiones y cómo han ido evolucionando hasta llegar a lo que estamos ahora. Para verlo de una forma sencilla lo he dividido en 5 grandes grupos que se note de uno a otro un paso importante y que hayan significado un antes y un después en este tema.

Veremos que en la última década toma un papel muy importante la informática al desarrollar cada vez más aplicaciones y diferentes algoritmos se pueden saber si las decisiones son más correctas o menos, acompañándonos de algo que está recientemente en auge como es la inteligencia artificial.

2.3.1 Fase 1

Las decisiones humanas antiguamente eran guiadas por la intuición, el humo, los sueños, los adivinos, los profetas y el famoso oráculo de Delfos. En el siglo sexto A.C. las decisiones se basaban en la benevolencia, en los rituales y en la reciprocidad, es

decir, se hacía lo que se debía hacer en cada momento y lo que se esperaba que se hiciese.

Hacía el siglo quinto A.C. se toman las primeras decisiones mediante votaciones, esto tiene lugar por primera vez en Grecia con las peonza de decisión ("discos para votar"). A continuación, se empieza a valorar la experiencia como fuente de información para toma de decisiones según el filósofo Aristóteles. 400 A.C., quinientos ciudadanos, toman una d las primeras decisiones de una tribuna de jurados: Condenar a Sócrates a la pena de muerte.

2.3.2 Fase 2

En el siglo IX, el sistema numérico indo-arábigo estimuló el desarrollo de las matemáticas y por ende de los modelos de probabilidad. En el siglo XIV, se priorizo el análisis de información para la toma de decisiones, es decir, se empezaba a analizar la situación y la información de la que se disponía antes de hacer la elección.

Surge el dilema más famoso de la literatura en 1602, que era una decisión, en la que sirvió para tener un largo debate entre los filósofos: "Ser o no ser".

En 1620 el razonamiento inductivo cobra importancia, consiste en considerar varias experiencias individuales para extraer de ellas un principio más amplio y general. Es importante tener en cuenta que, a pesar de que se parta de premisas verdades, la elección puede ser falsa. Obtener como conclusión una verdadera es apenas una probabilidad, cuyo grado varía de acuerdo al número de premisas que se consideren y a las características de éstas. Se trata de las decisiones con los algoritmos lógicos estudiados también a lo largo del grado que he cursado.

En 1641, Descartes propone que la razón es superior a la experiencia, trece años después Pascal desarrolla el concepto de cálculo probabilístico. En 1738, se establecen las bases de las ciencias del riesgo por Daniel Bernoulli.

Se introduce el concepto de regresión a la media para análisis de compra de acciones y negocios, entra en papel el sector laboral y la implicación que tienen las elecciones.

Como se puede ver en esta fase, damos un paso importante después D.C. en el que los filósofos toman un papel relevante en este tema y se van avanzando con los algoritmos implementados lógicos y los teoremas.

Ahora vamos a pasar a la fase 3 dónde entro en el siglo XIX y XX y veremos como los matemáticos establecen una serie de algoritmos y va avanzando poco a poco hacia lo que actualmente tenemos.

2.3.3 Fase 3

Siglo XIX, se desarrolla la campana de Gauss, una herramienta empleada en probabilidad y estadística.

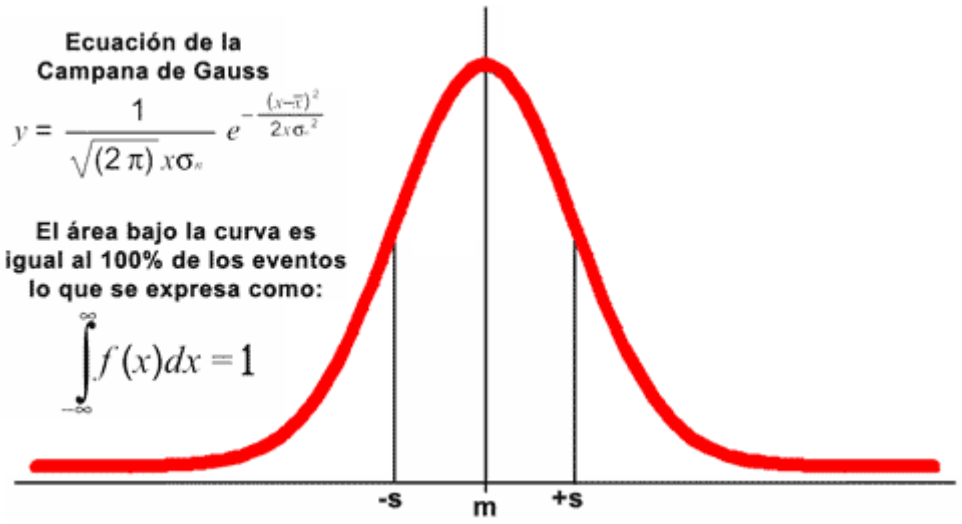


Ilustración 1: Ecuación de la Campana de Gauss

En 1900, Freud sugiere que las acciones y decisiones están influenciadas por el inconsciente. En muchas ocasiones sin darnos cuenta nos dejamos llevar por los impulsos sin analizar la información y la situación. En 1921 se hace la distinción entre riesgo e incertidumbre.

En 1947, nace la expresión “racionalidad restringida” para referirse a las decisiones que se toman sin tener la información necesaria y puede que bajo un presupuesto cerrado. Como bien dice la expresión, se piensa y se analiza pero hay que restringirse a la información que tenemos.

Vista la visión general de los siglos de esta fase, a mediados del siglo XX entra en juego la tecnología de mano de los computadores y el uso de ellos para algunas tomas de decisiones por lo que vamos a entrar en la era de la informática, la cual voy a resumir en una breve pasada en la siguiente fase.

2.3.4 Fase 4

Se empiezan a introducir las primeras herramientas de apoyo a la toma de decisiones basadas en computadoras. Se introduce el “Teorema de la Imposibilidad”, según el cual no puede existir un conjunto de reglas para la toma de decisiones que satisfaga todos los requerimientos de la sociedad.

En los años 60, se desarrolla la matriz DAFO, útil para tomar decisiones estratégicos.

Ilustración 2: DAFO

En 1965, se realizan las primeras investigaciones sobre la especialidad funcional de cada hemisferio cerebral. Siete años después, se acuña e término “groupthink”, es decir, se empieza a tratar las decisiones en grupo para poder llegar a un consenso.

En 1979 se demuestra que el modelo racional de la economía falla cuando las personas enfrentan incertidumbres en vida real, por lo que hay que buscar otra metodología diferente.

Dada la importancia que la tecnología toma en esta época, IBM se convierte en el principal proveedor de equipos tecnológicos para las empresas.

Se descubre que los ejecutivos a menudo combinan la planificación rigurosa con la intuición ya que no todo está en la información y análisis de datos y hay que valorar la intuición que te da en ese momento y a veces es conveniente dejarte llevar.

En 1989 se introduce el término inteligencia de negocios. Seis años después se desarrolla la prueba de asociaciones implícitas para detectar que actitudes o creencias inconscientes puede influenciar los juicios. Las personas comienzan a tomar decisiones de compra en la red, basándose las decisiones de otros usuarios.

En 2005 se explora la posibilidad de que las decisiones a veces son mejor las instantáneas que las panificadas por lo que habían estudiado antes de dejarse llevar por el impulso.

Para finalizar esta fase voy a sacar una serie de conclusiones que hay que tener en cuenta a la hora de tomar decisiones:

- El riesgo es una parte inevitable de una elección.
- El consenso entre una o más personas es bueno a no ser que sea una decisión muy sencilla.
- Las herramientas tecnológicas ya he mencionado que empiezan a tener importancia y son claves para la ayuda a la hora de elegir en el proceso establecido.
- Siempre se debe tener en mente la intuición.

Por poner algún caso de como entra la tecnología en este sector veamos casos específicos para hacernos una idea mejor.

Una evidente y que es el pan de cada día de todos nosotros es la *World Wide Web (www)* y los multimedia que tienen el potencial de ampliar el acceso a todo el mundo y

llegar desde cualquier lugar a la información necesaria para complementar nuestro contenido y poder tomar la decisión.

Los especialistas ofrecen a los directivos dos nuevas tecnologías completamente diferentes para los procesos administrativos (2000). Estas son la Ingeniería de Procesos y la Administración del Conocimiento, son dos técnicas que explican que el conocimiento con las tecnologías es fundamental en la toma de decisiones. La primera trata de la coordinación estructurada entre personas y es de arriba hacia abajo. En cambio, la segunda, se centra más en la eficacia que en la eficiencia y es de abajo hacia arriba.

Otro ejemplo, el paquete de *Microsoft Office* nos ha ayudado a poder contar con una herramienta para llevar un registro y un control de la información y en una hoja *Excel* poder contabilizar cada uno de los atributos que voy a valorar y compararlos uno a uno su peso para poder llegar a la mejor decisión.

2.3.5 Fase 5

En esta última fase voy a exponer el momento en el que nos encontramos, una vez ya nos hemos hecho una idea de los años pasados.

Ilustración 3: Evolución app móviles

Como cabe esperar, una vez ya he mencionado la tecnología como entra en nuestras vidas y como avanza de una manera descomunal, en estos últimos años lo más sacado son aplicaciones, por lo que no cabe duda que hay algunas para este tema que iré exponiendo ahora.

"Hey!" Es una aplicación para teléfonos móviles que tiene por misión generar una red personal para cada usuario, donde el usuario incluya a las personas que considera necesarias para su decisión y de esta forma crear un ecosistema de consejeros. De tal forma, que el usuario genere preguntas y obtenga respuestas de manera instantánea para tomar la decisión o pasado un tiempo que será determinado por el propio usuario.

"Bonus track" Es una aplicación para decisiones más simples que la anterior pero también necesarias. Se trata de una galería de fotos inteligente que te agrupa las fotos según temática y te elije la mejor foto de cada ocasión. La app promete organizar su red de archivos de fotos.

"Womity" Aplicación muy parecida a la primera, que puedes hacer en el móvil o desde el ordenador con tres simples pasos, en primer lugar defines que es lo que vas a decidir, y posteriormente metes los factores que influyen en la decisión y dejas que voten tus red de contactos de esta forma se hará una estadística para saber cual sería la mejor solución.

"ChoiceMap" Una vez descrita la decisión a tomar e ingresadas todas las alternativas que podrían darse, la app le permite explicar todos los factores que influyen en tu pensamiento y darles un peso que consideres oportuno para cada ocasión. Los resultados serán mostrados en una gráfica de barras que se puede compartir en Twitter o Facebook con tus amigos.

"Decision Buddy" Es igual que la anterior pero incluye varios extras que hacen que se diferencia. Incluye la posibilidad de toma de decisión por grupos de tal forma que los factores serán elegidos y ponderados por muchos miembros. Ambas son gratuitas para teléfonos Android.

2.4 Propuesta de solución

En este apartado concretaré las tecnologías seleccionadas como solución para el desarrollo del proyecto así como el análisis previo. Paralelamente se expondrá la solución del proyecto a exponer tras haber valorado el estado del arte y el contexto actual en el que nos encontramos.

Una vez analicé el marco en el que nos encontramos y tras la explicación de mi tutor, Antonio Berlanga del problema que existía, que todavía ninguna aplicación era capaz de cubrir todas las necesidades me dispuse a crear una aplicación web para la implementación del algoritmo de decisión multicriterio de tal forma que me sirva de ayuda para tomar una elección tanto a nivel personal como profesional dentro de los departamentos que lo necesiten.

Antonio me expuso el siguiente problema que os voy a mencionar seguidamente para que entendáis el porqué de esa necesidad en el mercado de aplicaciones web y como competencia muy importante de las que ya existen.

En un organismo público de Madrid todos los meses que quieren saber que barrio es en el que, por ejemplo; más robos se producen o cuál es el barrio que menos jóvenes van a la universidad o cuál es el barrio en el que más consumo de agua y luz se producen; tiran de una empresa externa que les hace una hoja Excel con los barrios de Madrid, los factores que influyen y se sacan en gráficas y listas ordenadas la solución a las preguntas planteadas.

El mayor inconveniente que encontré en ese planteamiento fue la pérdida de tiempo por parte de la empresa externa de cada vez que quieren un estudio nuevo deben hacer una hoja Excel desde cero y poner cada factor y en cada celda aplicar el algoritmo necesario.

Otro de los problemas es la forma visual de construir estas gráficas o listas ordenadas como resultados que solo eran capaces de hacer los empleados de la empresa externa ya que requería de conocimiento técnicos sobre el algoritmo para poder aplicar la cuenta en cada celda.

A la dificultad de uso y especialización a un sector determinado anterior, debemos sumarle lo poco visual e intuitivo que es la interfaz, porque no hay interfaz, partes de

una hoja en blanco. Para mostrar los resultados hay que pintarlo uno a uno en la gráfica o seleccionar las celdas que deseas pintar en una gráfica.

En resumen, nos encontrábamos bajo una demanda clara por parte de este organismo y pensé que esto no es una necesidad de solo ellos sino que todas las empresas lo reclamarían ya que es algo muy útil y rápido.

Como conclusión, tenemos un proceso de análisis y elaboración de los resultados y gráficas muy poco visual no tiene ningún tipo de experiencia de usuario, no es usable ni escalable, por lo que no hay mejor manera de solucionarlo por los tiempos que corren que con una aplicación web que esté al alcance de todas las empresas para facilitarles la toma de decisión y análisis de sus estadísticas dependiendo del mercado al que vayan dirigidas.

Mi propuesta a lo largo de todo este documento es la del desarrollo de una herramienta web para la resolución de problemas multicriterio con AHP. Se resolverán los problemas anteriores que he ido mencionando más extras que harán de esta aplicación algo muy atractivo y polivalente.

Ahora cada vez que tengan que hacer un estudio externo aplicando dicho algoritmo para poyar la decisión tienen a su disposición una aplicación que en tres simples pasos montan el árbol de decisión, los candidatos a la elección y por último muestran los gráficos y estadísticas de lo que encaja más con tus factores establecidos.

Para eliminar la sensación de poco visual y poco intuitivo, creo una interfaz en angularjs que está en auge en el mercado actual y que es una de las mejores tecnologías para desarrollo de aplicaciones web en la parte front-end del proceso.

Para dar más consistencia al proyecto he incluido algunas librerías que me permiten tener una guía de estilos predefinida, como es el caso de Bootstrap o Angular Material, que cuenta de un UI Kit muy básico pero necesario para este tipo de proyectos.

Como extras:

1. La importación de un proyecto que ya hayas realizado y no tienes que construir el árbol desde cero, es reutilizable.

2. Intuitivo. Te permite ver todo en tiempo real de tal forma que puedes interactuar con el árbol de decisión directamente con un "*drag and drop*" e ir viendo cuál es la estructura y los factores que se adaptan a tu decisión.
3. Diseño simple y claro, logrado con las librerías que he citado anteriormente (Bootstrap y Angular Material).
4. Gracias al estudio previo y análisis de los prototipos se ha logrado una experiencia de usuario que antes no se garantizaba.

Una vez expuestos los problemas que se van a solucionar con la creación de la aplicación y los extras que se van a incluir para diferenciarse dentro del mercado voy a pasar al siguiente apartado de tecnologías para ver más en detalle las tecnologías que se han implementado y cómo funcionan.

2.5 Tecnologías

ANGULARJS

AngularJS es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una página. Tiene por objetivo aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para que el desarrollo y las pruebas sean más fáciles.

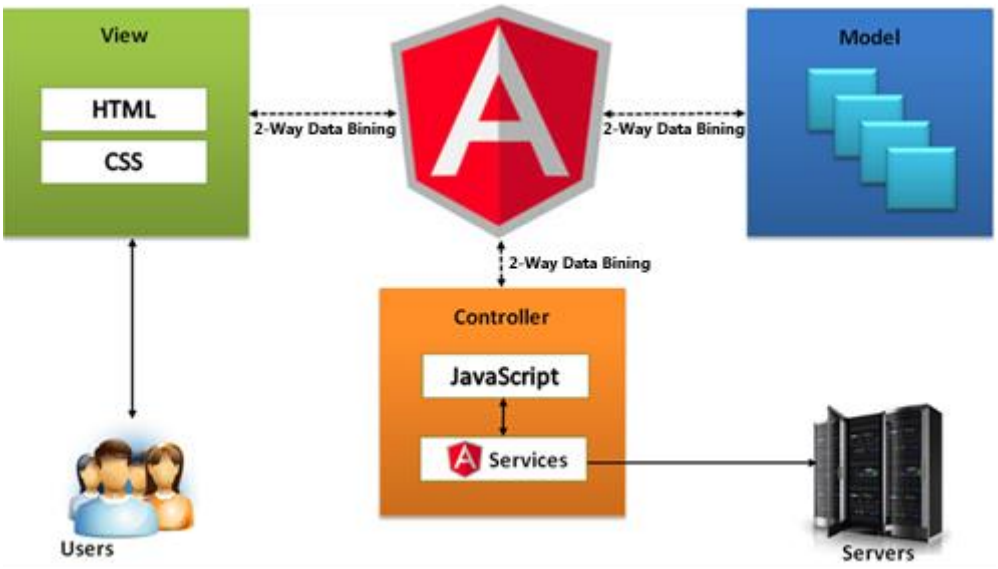


Ilustración 4: Modelo vista controlador

Se ha convertido en el framework JavaScript más potente y usado en aplicaciones web y móvil que dependen fuertemente en el frontend. Se trata de crear una estructura para crear aplicaciones web, a base de extender las capacidades HTML.

Con tanta oferta de framework se me hizo difícil elegir cuál usar en mi aplicación, a continuación voy a indicar qué ventajas tienen unos frente a otros y ahí se verá el porqué me decante por este.

Anteriormente en la parte Front-end de las aplicaciones web sólo teníamos a jQuery para ayudarnos con el código JavaScript del cliente. Podíamos manipular el DOM de una forma sencilla, añadir efectos y llamadas AJAX, pero no teníamos un patrón a seguir. Todo el código JS iba en funciones que íbamos creando según necesitáramos, lo que conlleva que con el tiempo el código sea difícilmente escalable y manejable.

Por suerte, fueron surgiendo nuevos framework que implementan el MVC y nos ayudan a separar conceptos. El más conocido BackboneJS, que surgió en 2010 y dependía de jQuery y UndersCore. Solían elegir este porque era la más conocida la que más seguridad daba y contaba con una documentación muy buena para empezar a implementar tu aplicación con ella.

Sin embargo AngularJS está pisando fuerte. Aunque su primera versión en 2009 tardó en hacerse popular 3 años y a partir del cuarto año empieza a estar en pleno auge. Tanto que se habla de una nueva "*technology stack*" como antes era ***LAMP*** (**L**inux + **A**pache + **M**ySQL + **P**HP) paso a ser la tendencia ***MEAN*** (**M**ongoDB + **E**xpressJS + **A**ngularJS + **N**odeJS).

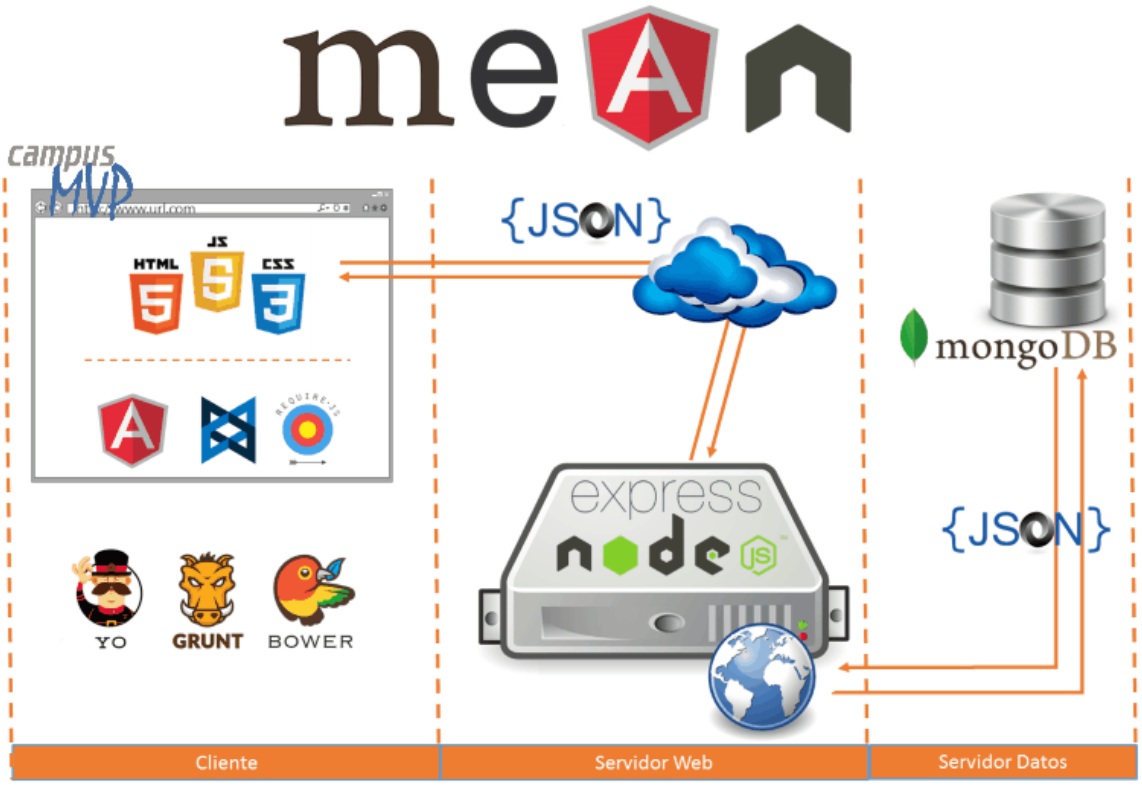


Ilustración 5: Arquitectura AngularJS

En la arquitectura montada para la aplicación se ha dejado preparada para incorporar una base de datos, como sería MongoDB, que en la imagen anterior es la parte de más a la derecha del servidor de datos.

El día de mañana que sea necesario una base de datos porque siempre se utilice para las mismas decisiones, es decir, siempre se utilice para elegir universidad, pues habría que crear la base de datos con todas las universidades y ahorraríamos al usuario introducir cada vez que va a tomar una decisión todas las candidatas que baraje, tiraría la aplicación directamente de la BBDD.

Otra situación en la que surgiría la necesidad de conectar con MongoDB; si el cliente al que vendemos la aplicación web la quiere personalizar para el departamento de recursos humanos, se conecta con la BBDD de la empresa con los candidatos y bastaría con introducir los criterios en cada caso para poder tener una decisión clara y en el menor tiempo posible.

Por estos casos que he comentado logro tener una aplicación flexible para cualquier caso, que aborda todos los factores y la hace realmente escalable, algo fundamental a la hora de integrar.

Otro punto a su favor es lo fácil que es crear tests unitarios y *End-to-End* con Karma, por ejemplo, algo que suele ser bastante tedioso sobre todo al principio.

Permite extender, como ya he dicho, el lenguaje HTML con directivas y atributos, manteniendo la semántica y sin necesidad de emplear librerías externas como jQuery o Underscode.js para que funcione.

Entrando en el comportamiento del código, con el objetivo de construir aplicaciones en cliente con HTML y JavaScript, es decir, con el peso de la lógica y el renderizado lo lleva el propio navegador de modo que hace la app mucho más fluida, en lugar de que la cargue el servidor, y se actualiza en tiempo real todos los ambos que se hacen en el código.

A alto nivel, y ahora los veremos uno a uno un poco más al detalle, tenemos:

- Plantillas HTML (***Templates***) con el markup de Angular.
- Escribo ***Componentes*** para gestionar las templates anteriores y ***Directivas*** que afectan al comportamiento de los componentes.
- Encapsulo la lógica de la aplicación en ***Servicios.***
- Defino ***Módulos*** que le dicen a Angular qué es lo que incluye la app, y cómo compilarlo para lanzarlo con *ng-module.*

Adjunto una imagen sacada de la web de Angular donde creo que se explica en el diagrama de una manera clara y muy visual la arquitectura típica

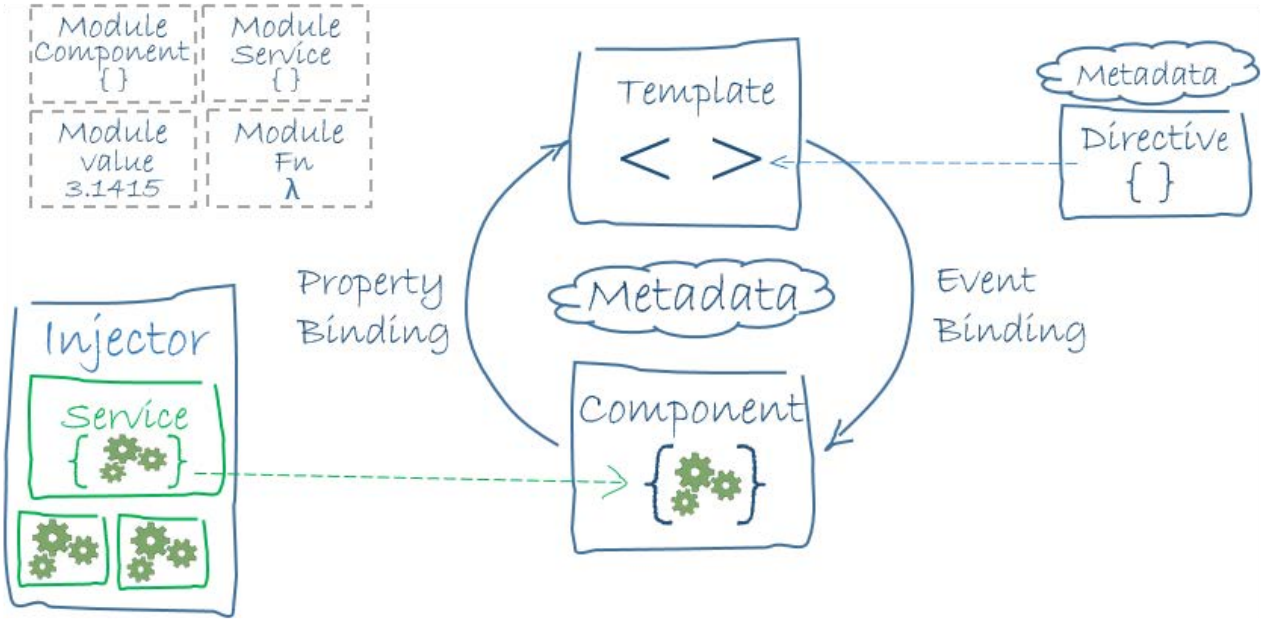


Ilustración 6: Diagrama de Angular

Módulos

Las apps de Angular son modulares, gracias a su propio sistema de módulos denominado ng-module. Toda aplicación debe tener al menos un módulo, el módulo principal, también conocido como *root module.*

Es un conjunto de código dedicado a un ámbito concreto de la aplicación, o una funcionalidad específica y se define mediante una clase *@NgModule.*

NgModule es un decorador que recibe un objeto de metadatos que será lo que define posteriormente el módulo. Los metadatos más importantes de este decorador son:

- **Declarations**: las vistas que pertenecen a tu módulo. Puede haber tres tipos de vistas: componentes, directivas y pipes.
- **Exports**: conjunto de declaraciones que deben estar accesibles para los templates de otros módulos de la aplicación.

- **Imports**: clases exportadas que son requeridas por templates de componentes de este módulo.
- **Providers**: Los servicios que son necesarios para el módulo, y que estarán disponibles para toda la app.
- **Bootstrap**: define la vista raíz, es decir, sólo se utilizará en el *root module* (llamado en el código, AppModule).

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent }  from './app.component';
import { SomeProvider }  from './some-provider';

@NgModule({
 imports: [ BrowserModule ],
 declarations: [ AppComponent ],
 bootstrap: [ AppComponent ],
  providers: [ SomeProvider ]
});
```

Componentes

Un componente controla una zona de la pantalla más conocida como vista. Es una clase estándar denominada *@Component.*

Define propiedades y métodos que están disponibles en su template, extrae toda la lógica en servicios para que el controlador solo se encargue de gestionar su vista.

Pueden tener atributos tanto de entrada ***Inputs()*** como de salida ***Outputs()*** para especificar los parámetros que entran o salen en cada caso del componente.

Templates

Es lo que te permite definir la vista de un componente, como si dijéramos "el HTML de la aplicación por cada parte", es decir, si tengo un componente de *dashboard* con su respectivo modelo, haría una template donde "pinte" lo que el usuario tiene que ver y con lo que va a interactuar. Si cuento además con un header, tendría un componente llamado *navbar* con su modelo y la template que lo "pinta", y así sucesivamente.

La única diferencia más llamativa con HTML puro, es que hay otras etiquetas que se pueden añadir para hacer referencia a componentes o directivas como son ***ng-model, ng-click, ng-component***.

Directivas

Se debe tener en cuenta que las templates de angular son dinámicas: cuando angular renderiza, transforma el DOM en base a las instrucciones de las directivas que encuentra.

Por asemejarlo a lo que ya he expuesto, un componente es un caso concreto de directiva que siempre va asociado a un template y al que por ser un elemento tan importante se le da posteriormente en Angular 2 un decorador propio, *@Directive.*

Hay otros dos tipos de directivas, las estructurales y las de atributo, que aparecerán normalmente como etiquetas de elementos HTML.

- **Estructurales:** en angularJS se separan por un guión y angular2 se diferencian fácilmente al comenzar por un asterisco, alteran el layout añadiendo, eliminando o remplazando elementos.
  - **ng-if**: si la condición se cumple se añade el elemento al DOM, en caso contrario, se elimina del DOM.
  - **ng-for:** repite su elemento una vez por cada ítem que exista en el iterador que se le pasa como parámetro.
- **Atributo:** alteran la apariencia o comportamiento de un elemento existente y su apariencia es igual a la de atributos HTML.
  - **ngClass:** esta directiva permite añadir y eliminar varias clases simultánea y dinámicamente. Por un lado, necesitaremos un método que gestione el estado de las clases CSS que queremos

aplicar, en un componente, y por otro, la template donde la vinculamos al método del controlador.

Servicios

Son imprescindibles, ya que en Angular se definen a través de simples clases. Todo valor, función o característica que mi aplicación necesita se encapsula dentro de un servicio.

Los componentes son los principales usuarios de servicios. No recuperan datos directamente del servidor, ni validan inputs de usuario, ni logean nada directamente en consola. Delegan todo este tipo de acciones en los servicios.

Estos objetos tienen métodos que sirven para mantener los datos en el ciclo de vida de la aplicación y se comunican a través de los controladores de manera consistente.

Es importante destacar que los servicios en AngularJS siempre son *Singleton*. Esto significa que una vez construyo un objeto servicio, la misma instancia es reutilizada, es decir, no hay dos instancias en un mismo servicio.

AngularJS viene ya con muchos servicios construidos que me permite hacer de la aplicación una herramienta consistente:

- **Constants:** una constante es un servicio al que le pasamos directamente el valor de dicho servicio.
- **Value:** servicio al que le pasamos directamente el valor de dicho servicio en forma de valor.
- **Service:** el servicio recibe un parámetro y es capaz de crear internamente una instancia de la clase.
- **Factorías:** es similar a un servicio pero más detallado y configurable, por lo que me proporciona libertad a la hora de determinar instancias.
- **Provider:** un proveedor es como una factoría pero permite que se configure antes de crear el valor del servicio.

ALGORITMO DE DECISIÓN MULTICRITERIO

En la toma de decisiones, es importante distinguir entre los casos en que se utilizan un solo criterio o varios. Cuando se utiliza uno, la decisión se toma determinando la alternativa que presenta la mejor puntuación para el criterio considerado. En este caso, utilizaría algoritmos basados en un criterio que puede ser de coste, beneficio, utilidad o satisfacción, entre otros.

Por la otra parte, si utilizo más de un criterio, se trata de una toma de decisión multicriterio, mi caso concreto. En inglés se denomina *Multiple Criteria Decision Making,* y para abordar este tipo se han definido un número muy alto de algoritmos como el Modelo de Suma Ponderada, Proceso Analítico Jerárquico, Modelos de Utilidad Multiatributo, entre otros.

Voy a explicar el algoritmo multicriterio que voy a implementar para mi aplicación web ***MasterSearch***.

Proceso Analítico Jerárquico

Fue desarrollado a finales de los 60 por Thomas Saaty, quien a partir de sus investigaciones en el campo militar y su experiencia docente formuló una herramienta sencilla para ayudar a las personas responsables de la toma de decisiones.

AHP es uno de los algoritmos de toma de decisión más utilizados. En este algoritmo, el problema de toma de decisión consiste en un objetivo global, un grupo de opciones o alternativas para alcanzar el objetivo, y un grupo de factores o criterios que relacionan las alternativas con el objetivo.

Su simplicidad y su poder han sido evidenciados en las cientos de aplicaciones en las cuales se han obtenido importantes resultados y en la actualidad, es la base de muchos paquetes de software diseñados para los procesos de tomas de decisiones complejas. Además, ha sido adoptado por numerosas compañías para el soporte de los procesos de toma de decisiones complejas e importantes.

Es un método matemático creado para evaluar alternativas cuando se tienen en consideración varios criterios y está basado en el principio que la experiencia y el conocimiento de los actores son tan importantes como los datos utilizados en el proceso.

Una vez construido el árbol de decisión que es el primer paso de la aplicación, con su determinada jerarquía y pesos, el algoritmo evalúa sistemáticamente los criterios y las alternativas comparándolos entre ellos con respecto a su impacto sobre el "elemento padre", es decir, las alternativas se comparan por pares con respecto a cada uno de los criterios y los criterios a su vez, se comparan por pares con respecto al objetivo.

Así, se asigna un peso relativo a cada comparación por pares de alternativas y criterios según la escala de a siguiente tabla:

Intensidad de importancia	Definición	Explicación
1	Igual importante	Los dos factores contribuyen igualmente al objetivo
3	Más importante	La experiencia y el juicio favorecen ligeramente a una opción sobre la otra
5	Mucho más importante	La experiencia y el juicio favorecen fuertemente a una opción sobre la otra
7	Mucho más importante	La experiencia y el juicio favorecen muy fuertemente a una opción sobre la otra
9	Absolutamente más importante	La evidencia favorece de manera indiscutible a una opción sobre la otra
2, 4, 6, 8	Valores intermedios	Útil en casos donde existe cierta dispersión de impactos

Ilustración 7: Tabla de pesos relativos algoritmo AHP

Cuando las prioridades de los elementos en cada nivel se tienen definidas, se agregan para obtener las prioridades globales frente al objetivo principal. Los resultados frente a las alternativas se convierten entonces en un importante elemento de soporte para quien debe tomar la decisión.

La notación utilizada es la siguiente:

- Para i objetivos dados $i = 1, 2,..., m$; se determinan los respectivos pesos w_i.
- Para cada objetivo i, se comparan las $j = 1, 2,..., n$ alternativas y se determinan los pesos w_{ij} con respecto al objetivo i.
- Se determina el peso final de la alternativa W_j con respecto a todos los objetivos: $W_j = w_{1j}w_1 + w_{2j}w_2 + ... + w_{mj}w_m$

Las alternativas se ordenan de acuerdo con el W_j en orden descendente, donde el mayor valor indica la alternativa más preferida. Las diferentes metodologías para la solución de problemas multicriterios se diferencian en la forma como determinan el objetivo y las ponderaciones a los factores.

Una vez conocidas las notaciones y la visión general del algoritmo es clave saber los principios y axiomas que fundamentan este algoritmo:

Principio de Descomposición

Intensidad de importancia	Definición	Explicación
1	Igual importante	Los dos factores contribuyen igualmente al objetivo
3	Más importante	La experiencia y el juicio favorecen ligeramente a una opción sobre la otra
5	Mucho más importante	La experiencia y el juicio favorecen fuertemente a una opción sobre la otra
7	Mucho más importante	La experiencia y el juicio favorecen muy fuertemente a una opción sobre la otra
9	Absolutamente más importante	La evidencia favorece de manera indiscutible a una opción sobre la otra
2, 4, 6, 8	Valores intermedios	Útil en casos donde existe cierta dispersión de impactos

Ilustración 8: Escala de comparación de Saaty

Los valores 2, 4, 6 y 8 se utilizan cuando no se puede definir con claridad la preferencia entre los factores. Estos son valores intermedios de preferencia.

Juicios Comparativos

Permite formar parejas de todos los elementos de un sub-grupo con respecto al criterio principal del subgrupo, por ello se habla de comparaciones biunívocas.

Composición Jerárquica

Permite producir prioridades globales a través de las multiplicaciones de las prioridades locales, es decir, que una vez se tienen soluciones locales, se agregan para obtener la solución general que se está buscando. Se va ascendiente desde los "nodos hojas" hasta el nodo padre global, identificado como objetivo.

Axioma recíproco

Si frente a un criterio, una alternativa A es n veces mejor que B, entonces B es 1/n veces mejor que A. Este principio es utilizado en el análisis matricial que se realiza a los criterios y las alternativas. Garantiza que el análisis se haga de manera bidireccional.

$$Aw = \begin{bmatrix} w_1/w_1 & w_1/w_2 & \cdots & w_1/w_n \\ w_2/w_1 & w_2/w_2 & \cdots & w_2/w_n \\ \vdots & \vdots & \cdots & \vdots \\ w_n/w_1 & w_n/w_2 & \cdots & w_n/w_n \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = n \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

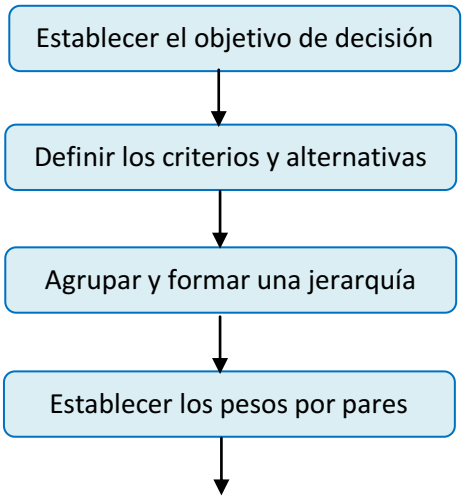
Donde $w_1 \ldots w_n$ son los pesos de las alternativas o criterios y w_1 / w_n son los pesos de las comparaciones de los criterios y de las alternativas con respecto a los criterios.

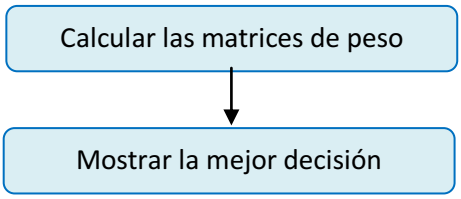
Finalmente, se calculan los pesos finales de las alternativas, que representan la capacidad de las alternativas para cumplir con el objetivo de la decisión, sumando los productos de los pesos de los criterios por los pesos de las alternativas con respecto a los criterios, como se hace en el Modelo de Suma Ponderada.

En este proyecto, he adaptado el algoritmo al entorno personal y profesional del día a día generalizándolo para cualquier tipo de decisión en el sector correspondiente. Para ello, he modificado dicho algoritmo en los siguientes aspectos:

- He utilizado intervalos en lugar de un solo valor en los pesos relativos de las comparaciones por pares para modelar las variaciones de los datos debidas a la incertidumbre que puede causar o el mismo estado de ánimo que influye a la hora de toma de decisiones. Para corresponder estos intervalos con los establecidos en la tabla anteriormente adjunta, se fijará de 0 a 100 el intervalo y cada rango definido corresponderá al número 1, 3, 5, 7 respectivamente.
- Las alternancias y los criterios se consideran variables en el tiempo y dependientes de cada persona que tome la decisión por lo que serán revisadas periódicamente en cada proyecto nuevo.

A continuación adjunto un esquema de los pasos que sigue el algoritmo para calcular las comparaciones por pares y llegar a la mejor elección.





Ejemplo de aplicación del algoritmo

Dado que la teoría siempre suena más compleja, voy a poner un ejemplo con las tablas que irían calculándose según los pasos del algoritmo una vez hecho el árbol de decisión y la comparación por pares desde abajo hacia arriba.

De esta forma me supone más ventajas a la hora de exponerlo ya que al ser complejo no se puede explicar con palabras y que quede claro.

El ejemplo, ha sido diseñado para trabajar el problema de la localización, el cual es multicriterio puesto que intervienen en esta decisión una gran variedad de criterios.

El problema consiste en seleccionar el mejor sitio para establecer un centro de atención de primer nivel que incluya los servicios de medicina general y odontología para cubrir la demanda de la población del norte del Cauca (Colombia) y se han definido tres alternativas de lugares, a continuación se presentan las características del proyecto y de las alternativas posibles:

Características del proyecto:

Población beneficiada: 15000 habitantes

Infraestructura física: Un local con 2 consultorios médicos, un consultorio para promoción y prevención y un consultorio odontológico, equipo de cirugía ambulatorio y ayudas diagnósticas y laboratorio clínico (rayos X). Área aproximada 200 m2.

Inversión: $120.000.000 (equipos, construcción y licencias)

Personal: 6 médicos generales (media jornada) 1 odontólogo, 1 enfermera jefe, 1 bacteriólogo, 1 auxiliar de enfermería y 1 auxiliar de laboratorio, 1 higienista oral, 1 recepcionista, 1 director médico administrativo, 1 auxiliar de oficina, 1 conductor (La ambulancia se tendrá en convenio con el hospital departamental del Valle del Cauca)

Jornada de atención: 7 a.m a 9 p.m lunes a sábado y 8 a.m a 1 p.m domingos y festivos.

Estos son los candidatos de localización:

Puerto Tejada

- Población objeto 3500 habitantes.
- Costo del m2 construido = $25.000
- Servicios de energía, agua, teléfono y alcantarillado.
- El agua debe ser procesada por ser de pozo costo $5000/m3
- Energía a $90/Kw
- Nivel de inseguridad alto. Presencia fuerte de guerrilla
- Acceso a la vía panamericana
- Estación de policía
- A 25 minutos del hospital de Santander de Quilichao (nivel 2)
- A 45 minutos del hospital departamental del Valle
- Servicio continuo de transporte para el Cauca y Valle

Caloto

- Población objeto 1500 habitantes
- Costo del m2 construido = $20.000
- Servicios de energía, agua, teléfono y alcantarillado
- El agua debe ser procesada por ser de pozo costo $6500/m3
- Energía a $100/Kw
- Nivel de inseguridad medio
- Acceso a la vía panamericana
- Estación de policía

- A 45 minutos del hospital de Santander de Quilichao
- A 120 minutos del hospital departamental del Valle
- Servicio periódico de transporte para el Cauca y Valle

Santander de Quilichao

- Población objeto 6500 habitantes
- Costo del m2
- construido = $35.000
- Servicios de energía, agua, teléfono y alcantarillado
- Energía a $80/Kw
- Nivel de inseguridad alto
- Acceso a la vía panamericana
- Estación de policía
- Hospital de segundo nivel
- A 60 minutos del hospital departamental del Valle
- Servicio continuo de transporte para el Cauca y Valle

A continuación, se deben establecer los criterios en los que nos basamos para tomar la decisión.

- Población: la mayor cobertura posible sobre la población objeto
- Cercanía al hospital: en caso de presentarse una emergencia no tratable en este centro de salud
- Costo del terreno: para ubicar el centro
- Seguridad: de la zona, tanto para los empleados como para los pacientes que demanden el servicio
- Servicios públicos: disponibles de la zona
- Transporte: se refiere a la disponibilidad de transporte para la zona

Una vez se han definido los criterios, se realiza el análisis por pares, es decir, se comparan cada una de las alternativas frente a cada uno de los criterios. Por ejemplo, si se quiere evaluar el criterio población, se tendría una matriz como la que se presenta en la tabla en la cual, el valor de 7 indica que se está prefiriendo muy fuertemente a Santander frente a Caloto y el valor de 5 muestra una preferencia fuertemente de Santander sobre Puerto Tejada, igualmente, el 3 indica que se prefiere moderadamente a Puerto Tejada sobre Caloto. De igual manera, los valores 1/7, 1/5 y

1/3 corresponden a los inversos lo cual explica que la diagonal corresponda a valores de 1, pues refleja la comparación del factor contra el mismo.

Criterio o factor:	**Población**		
Candidatos	Santander	Caloto	Puerto Tejada
Santander	1	7	5
Caloto	1/7	1	1/3
Puerto Tejada	1/5	3	1

Tabla 1: Matriz de comparación del criterio de Población

Después de haber realizado las comparaciones de todos los factores, estas matrices son normalizadas, es decir, se divide cada término de la matriz sobre la suma de sus columnas el resultado se muestra en la tabla 2.

Criterio o factor:	**Población**		
Alternativas	1	2	3
Santander	35/47	7/11	15/19
Caloto	5/47	1/11	1/19
Puerto Tejada	7/47	3/11	3/19

Tabla 2: Matriz normalizada del criterio Población

15/217
1/12
17/88

Tabla 3: Vector de prioridad del criterio población

Con esta matriz, se obtiene el vector de prioridad del criterio al promediar los valores de las filas. Este procedimiento se repite para todos los criterios y también se realiza para comparar los criterios entre sí.

	1	2	3	4	5	6
Población	1	5	7	2	1/2	1
Distancia hospital	1/5	1	3	1/5	1/5	1/5
Costo terreno	1/7	1/3	1	1/5	1/5	1/5
Seguridad	1/2	5	5	1	1/3	5
Servicios públicos	2	5	5	3	1	7
Transporte	1	5	5	1/5	1/7	1

Tabla 4: Matriz de comparación de criterios

	1	2	3	4	5	6
Población	70/339	15/64	7/26	10/33	105/499	5/72
Distancia hospital	14/339	3/64	3/26	1/33	42/499	1/72
Costo terreno	10/399	1/64	1/26	1/33	42/499	1/72
Seguridad	35/339	15/64	5/26	5/33	70/499	25/72
Servicios públicos	140/339	15/64	5/26	5/11	210/499	35/72
Transporte	70/339	15/64	5/26	1/33	30/499	5/72

Tabla 5: Matriz normalizada

Con cada vector de prioridad obtenido para los criterios, se conforma una matriz de prioridad la cual se multiplica matricialmente con el vector de prioridad obtenido al realizar la comparación entre los criterios.

Criterio	**Prioridad**
Población	203/942
Distancia hospital	13/235
Costo terreno	13/368
Seguridad	128/657
Servicios públicos	62/169
Transporte	76/575

Tabla 6: Vector de prioridad de criterios

El resultado, es un vector denominado vector de prioridad de las alternativas, el cual se constituye en la solución del problema, al presentar cada una de las alternativas y un porcentaje de preferencia para cada una de ellas.

	Alternativas	**Prioridades**
Santander	1	52,45%
Caloto	2	23,53%
Puerto Tejada	3	24,02%
		100,00%

Tabla 7: Vector de prioridad de las alternativas

Según esto, la mejor opción es Santander de Quilichao con un 52,45% y quedan en iguales condiciones de comparación las otras dos alternativas. Es claro, que el método propone una solución, pero quien finalmente toma la decisión es la persona o grupo encargado de hacerlo.

**Esta información corresponde a un documento de la Universidad Tecnológica de Pereira, documento citado en la bibliografía.*

NODEJS

Es una librería y entorno de ejecución de E/S dirigida por eventos y asíncrona, se ejecuta sobra JavaScript y ha sido creado por Google V8.

Proporciona un entorno de ejecución del lado del servidor que compila y ejecuta JS a grandes velocidades. La ventaja que tiene es que posee un excelente modelo de eventos, lo que proporciona una herramienta rápida, a la hora de desarrollo y de ejecución de test unitarios, todo esto hace que la experiencia de usuario sea más satisfactoria.

BOWER

Bower es un complemento ideal para el desarrollo web. Es un sencillo programa que nos sirve para tener al día las dependencias de un proyecto para la web, en lo que

respecta al desarrollo frontend, con Javascript o incluso CSS. Se trata de un programa basado en NodeJS que se ejecuta desde la consola y que tiene un sencillo API de comandos útiles para realizar tareas de mantenimiento y administración de paquetes necesarios para construir un proyecto web, concretamente la parte del lado del cliente.

Con Bower podemos descargar y actualizar todo tipo de librerías, frameworks, plugins, etc., pero sin tener que preocuparnos por descargarlos y subirlos a mano. Con esto quiero decir que no es obligatorio utilizar bower en un proyecto, pero es más cómodo. ¿Por qué? Porque de esta forma cuando he querido meter angular material para utilizarlo en la definición de los diseños y seguir un patrón, directamente lo hago con este comando:

bower install angular-material –s

Si no contase con esta herramienta tengo que ir a la página de angular material encontrar la versión adecuada descargar e importarla en el proyecto para poder utilizarla en los componente que desarrollo.

¿Cómo instalar bower?

Es un programa realizado con NodeJS, este es el principal motivo para instalarlo en primer lugar como he citado en el apartado superior. Una vez hemos instalado nodejs para instalar bower:

npm install –g bower

Es bueno tener un archivo bower.json en la carpeta raíz del proyecto. De manera que aquí se guardarán las dependencias que tiene el proyecto.

bower init

El json sería una cosa así: **Ejemplo sacado de desarrolloweb.com*

```
{
  "name": "Prueba Bower",
  "version": "0.0.0",
  "authors": [
    "Miguel Angel Alvarez "
  ],
  "dependencies": {
    "jquery": "~2.1.4",
    "bootstrap": "~3.3.5"
    "angular": "1.4.7",
    "angular-route": "1.4.7",
  }
}
```

KARMA

Se encarga de ejecutar test unitarios de JavaScript según se vaya construyendo el proyecto, de tal forma que ante cualquier fallo se detecta rápidamente y se puede ir cambiando en tiempo real para ir añadiendo valor al proyecto en cada iteración.

Para instalarlo es necesario nodejs y bower, simplemente con ejecutar el siguiente comando en la consola se crea en el directorio un archivo karma.conf.js para crear los test unitarios.

bower install karma –s

ANGULAR MATERIAL

Es una biblioteca de componentes de interfaz de usuario que me proporciona un patrón de diseño que permite desarrollar la interfaz de una manera consistente y rápida basándome en el diseño de cada uno de los componentes.

Para instalarlo simplemente hay que escribir en consola:

bower install angular-material –s

BOOTSTRAP

Son un conjunto de herramientas de código abierto para el diseño de aplicaciones web, contiene plantillas de diseño, formulario de botones, cuadros, menús… y está basado en HTML y CSS. Es modular y consiste en una serie de hojas de estilo LESS que implementan la variedad de componentes de una herramienta.

Es similar a angular material pero proporciona variedad respecto a componentes, por lo que cuento con las dos bibliotecas para que el abanico de elección sea más grande y perseguir el objetivo de aportar una aplicación más consistente de gran valor.

3. Entorno de desarrollo

3.1 Introducción

Una vez detalladas en el capítulo las tecnologías que empleo y las librerías para el desarrollo de la app web. Aquí es el momento de exponer las características de Hardware y Software que he utilizado en el estudio, realización y ejecución de proyecto.

Incluiré un manual de configuración inicial de como arrancar el proyecto, las instalaciones que he seguido y el orden de ejecución.

3.2 Hardware

Como la aplicación web se ejecuta en el local host en el puerto 3000 he utilizado el mismo hardware para el desarrollo como para la ejecución.

Ordenador central dónde se desarrolla el código e implementa a aplicación:

Fabricante: Apple

Modelo: MacBook Pro (Retina, 15 pulgadas)

Procesador: 2,2 GHz Intel Core i7

Memoria: 16GB 1600 MHz DDR3

Gráficos: Intel Iris Pro 1536 MB

Para la redacción de la memoria he utilizado una Microsoft Windows Surface Pro ya que contaba con Windows y podía disponer de Office a la vez que veía la aplicación en el otro monitor:

Fabricante: Microsoft Windows Surface

Modelo: Surface Pro 2017

Procesador: Intel Core i7

Memoria: 1TB de almacenamiento

Sistema operativo: Windows 10 Pro

Pantalla: 12,3” PixelSense de alta resolución

3.3 Software

En la siguiente sección expongo los elementos de software que he necesitado para poder llevar a cabo el proyecto. Incluyo el software total tanto para redactar el proyecto como para implementar el código y su correspondiente ejecución.

3.3.1 Sistema operativo

Como he comentado en el apartado de hardware he utilizado dos ordenadores con diferentes sistemas operativos:

Windows 10

Sistema operativo instalado en el ordenado Surface Pro para redactar la memoria de este proyecto.

MacOS

Es el sistema operativo detrás de cada Mac. Te permite hacer cosas que simplemente no puedes hacer con otras computadoras porque está diseñado específicamente para el hardware en el que está instalado, y viceversa.

3.3.2 IDE

Visual Studio Code

Es un editor de código fuente ligero no un IDE completo, pero potente que se ejecuta desde el escritorio y está disponible para todos los sistemas operativos, en mi caso macos. Viene con soporte incorporado para JavaScript, TypeScript y Node.js.

Una de las cosas más potentes de este editor es su gratuidad y que es multiplataforma. Está disponible para todos los sistemas operativos y implementa varios lenguajes de programación, hoy en día es raro que no cubra la necesidad que necesitas.

Tiene integración con Git simplificada para que los usuarios pueda trabajar con este entorno colaborativo de forma muy sencilla, publicando ramas, push y pull request.

Otra de sus características es su fácil adaptación gracias a la simpleza y al buen hacer de los diseñadores que trabajaron en la experiencia de usuario, “no hay que utilizar manuales de mil páginas”.

Microsoft destaca por su documentación, gran ventaja a su favor.

Como dato ya más personal es que te permite personalizar la interfaz para que esté a mi gusto y trabajes más cómodo.

Sketch

Aplicación del diseño vectorial pensada para diseñadores, cuenta con capas, gradientes, ruido, desenfoque y permite diseña pixel por pixel.

Es una herramienta; como he mencionado, para diseñadores, que he empleado para hacer los diseños de la interfaz y poder crear una aplicación web con consistencia y coherencia dentro del patrón angular material y Bootstrap que he instalado como librerías.

Elegí Sketch porque llevo seis meses como UX en una consultar de Big Data y he cogido manejo con la herramienta, tengo disponibilidad de licencias y es ahora mismo el auge en este mundo ya que cuenta con un montón de plugins para hacer diferentes cosas

Uno de los plugins y muy interesante es el craft de InVision con el que puedes realizar transiciones entre pantalla de manera muy sencilla animas los “artboards” haciendo de in diseño algo “iterativo”

Draw.io

Herramienta que nos permite hoy en día elaborar diagramas en línea sin necesidad de instalar nada en el ordenador. Tiene una interfaz bastante sencilla y fácil de utilizar.

No me ha hecho falta un software de pago ya que este te permite hacer todo lo que he necesitado. Sirve para hacer cualquier diagrama de clases, de flujo necesarios para un proyecto software de este calibre.

Node.js

La meta número uno declarada de Node es proporcionar una manera fácil para construir programas de red escalables.

Resuelve este problema cambiando la forma en que se realiza una conexión con el servidor. En lugar de general un nuevo hilo de OS para cada conexión (y de asignarle la memoria acompañante), cada conexión dispara una ejecución de evento dentro del proceso del motor de Node.

No permite bloqueos y se bloquea directamente para llamados E/S. Un servidor que lo ejecute puede soportar decenas de miles de conexiones concurrentes.

Bower

No pertenece a un entorno de desarrollo como tal, pero dentro del hardware que he utilizado considero que en la mejor sección que encaja es en esta ya que no es una librería ni mucho menos un sistema operativo.

Se trata de un gestor de dependencias para la implementación de una app web front que me facilita la tarea de instalación de los diferentes proyectos y frameworks.

Es un programa basado en NodeJS que se ejecuta desde la consola y que se utiliza con unos comandos muy sencillos útiles para tareas de mantenimiento y administración de paquetes necesarios para construir mi proyecto.

3.3.3 Librerías

Angular-ui-tree

Es un componente AngularJS UI que te permite ordenar listas anidadas, proporciona soporte para hacer "*drag and drop*" y no depende de Jquery.

Sus características principales:

- Son que utiliza el ámbito AngularJS nativo para la vinculación de datos.
- Ordena y mueve elementos con la propiedad de manera visual como si fuera un árbol.
- Evita que los elementos acepten nodos secundarios.

Angular-material

Es un marco de componente de interfaz de usuario y una implementación de referencia de la especificación de diseño de material de Google.

Esta librería ejerce como proyecto, de manera que me proporciona un conjunto de componentes de interfaz de usuario reutilizables, testeados y accesibles basados en el diseño del material.

Cuando quiero utilizar un componente de esta librería accedo al siguiente enlace: https://material.angularjs.org

Bootstrap

Framework gratuito de front para facilitar el desarrollo y su implementación. Incluye platillas de diseño basadas en HTML, CSS para tipografía, formularios, tablas y muchos más componentes.

Te permite añadir complementos adicionales de Java Script así como crear diseños de respuesta lo más rápidos posibles.

En este proyecto yo he implementado la anterior (angular-material) para componentes y Bootstrap para los colores de los botones dependiendo si son de acción, de información, utilizo una clase u otra.

No mezclo un componente con otro de las librerías porque rompe la consistencia, aunque ambas las escogí porque son muy similares la línea que tiene de experiencia de usuario, sencilla y limpia. Todo esto da una sensación de sencillez y de aplicación atractiva, “sin necesidad de manual”.

JQuery

Librería de JavaScript desarrollada para facilitar el desarrollo de código JavaScript. Ofrece diferentes funcionalidades para facilitar la interacción con el código HTML y el árbol de ui tree.

Me da la facilidad de cambiar los estilos de los elementos, añadir efectos, funcionalidades o incluir eventos del usuario.

Además es una librería que ofrece funciones AJAX que permiten hacer llamadas asíncronas al servidor, lo que te permite cargar pequeñas partes de información sin necesidad de cargar la página entera.

Se puede descargar desde la página oficial de JQuery, es necesario descargar la de JQuery UI que van ligadas. Me descargo la versión que sea compatible con todas las otras.

3.4 Manual de configuración

Las bases del proyecto son tan importantes como todo lo demás por eso he de añadir un apartado de manual de configuración "starter" para saber cómo empezar un proyecto de una aplicación desde 0, y cómo lanzarlo cada vez que vas a editarlo.

Esta decisión la tome, porqué al terminar la carrera cuando te afrontas a crear una aplicación desde cero, resulta un poco complicado así como definir una arquitectura del proyecto por eso considero importantes tener dichas referencias para un caso futuro.

Instalación y configuración del entorno

1. Descargar Visual Studio Code
2. Abrir la aplicación
3. Personalizar la aplicación para ver los archivos en su extensión.
4. Generar un proyecto con la estructura habitual de angularjs. Para partir de un árbol de directorios lógicos y propio de esta tecnología.
    a. Acceder a ***yeoman.io/generators/***
    b. Seleccionar angularjs
    c. Descargar el proyecto generado
    d. Guardar en un directorio localizado
    e. Copiar el directorio
5. Abrir directorio del proyecto desde la aplicación Visual Studio que tenemos abierta.
6. Eliminar la sobre programación de carpetas o ficheros generados por yeoman.io que no son necesarios para el proyecto.
7. Entrar en ***nodejs.org/es*** para descargar el entorno de ejecución para npm y bower.
8. Instalar desde la consola del Visual Studio Code el paquete de nodeJS que acabo de descargar.
9. Una vez se haya instalado para npm ejecutar el siguiente comando:

npm install –g bower

10. Una vez instalado el bower me va a generar en el árbol de directorios un archivo ***bower.js*** con las dependencias y versiones de los paquetes o librerías instaladas en el proyecto con este gestor de paquetes.
11. Con el gestor instalado procedo a instalar las librerías necesarias para agilizar el desarrollo.
12. Instalar angular-ui-tree, librería para pintar la lógica de un árbol:

bower install angular-ui-tree --save

13. Instalar angular material, para contar con los componentes y el diseño realizado por Google con esta librería.

bower install angular-material --save

14. Instalar Bootstrap para colores y tipografías.

bower install bootstrap --save

Una vez se ha realizado la configuración e instalación del proyecto, está listo para desarrollar así que solo tengo que ir incluyendo controladores, servicios, vistas y modelos.

Desde la consola cuando se abre Visual Studio Code ejecuto el siguiente comando:

gulp serve

Si cuento con dos pantallas simultáneas tengo en un lado el código y en otro la aplicación abierta en el navegador en el puerto 3000 con la url: ***localhost: 3000.***

Cada vez que guarde un cambio nuevo como estamos trabajando del lado del cliente el navegador se actualiza automáticamente y puedo ver en tiempo real los cambios y como que la interfaz directamente.

3.5 Conclusión del capítulo

En este capítulo queda explicada y descrita de forma detallada la instalación y configuración del entorno de desarrollo que forma el proyecto.

Sirve de guía inicial para que en un futuro cuando se dé un caso así sepa el autor del proyecto por dónde empezar, con la guía incluida y como lanzarla cada vez que realizo una modificación o incluyo un cambio en el versionado del código.

4. Análisis de sistema

4.1 Introducción

A lo largo de este capítulo se definirán las directrices del sistema a desarrollar. Tiene como objetivo dar una visión detallada (al margen de la maquetación y diseño) de la funcionalidad completa a desarrollar para llegar hasta la "isla ideal" que sería el sistema demandado, el requisito global del proyecto.

Una vez realizado el estado del arte actual, la descripción del proyecto y sus tecnologías, nos encontramos centrados en el contexto del proyecto por lo que voy a entrar a esta siguiente fase de análisis.

En esta fase se recoge la información correspondiente a la consecución de los casos de uso y requisitos de software.

Para la elaboración de estos requisitos y casos de uso se debe tener en cuenta los stakeholders del proyecto. En este caso concreto el proyecto va destinado de forma muy genérica, abarcar todo el mercado, ya que siempre toman decisiones.

Siempre hay que decantarse por poner foco, como dice la metodología agile, en una cosa para aportar más valor al producto. Lo pongo en los ayuntamientos para

estudio y en equipos de RRHH, es decir centrado para un usuario final como empresa, en lugar de personal y más individual. Una de las razones también de esta elección es que está pensado para ser utilizada en un ordenador no en un móvil dónde la utilizaría cualquier persona en su casa para decidir el coche que se va a comprar, por ejemplo.

4.2 Estudio de la propuesta

Se ha solicitado el desarrollo de una aplicación que implemente el algoritmo multicriterio basado en AHP para cambiar el modelo tradicional que utilizan hoy en día para implementarlo.

Ahora mismo la forma de utilizar dicha herramienta es mediante una hoja excel que cada uno se configura para que al rellenar una matriz en las celdas consecutivas se calcule la suma ponderada y determinadas acciones para llegar al final del algoritmo.

El caso real de dónde surgió la necesidad es del Ayuntamiento que lo utiliza para saber que barrios son más conflictivos, más ricos, el alquiler más barato, e infinitas cosas que se pueden ver con este algoritmo de decisión.

Para ello, se desarrollará un software como aplicación web que sea capaz de forma simple y muy visual plasmar el algoritmo y cuando tenga el usuario o la empresa todos los criterios metidos pueda obtener la lista de una manera sencilla. La aplicación recibirá el nombre de Master Search, dado que es un algoritmo de búsqueda de la mejor decisión.

Se debe tener en cuenta que además de los requisitos del cliente, se aporta mucha funcionalidad desde la parte de desarrollo para, de una forma ágil con la metodología, aportar valor en cada iteración.

El alcance de esta aplicación puede ser muy grande como impacto en el mercado como alternativa y posiblemente sustituto del modelo tradicional. Esto se debe al ahorro de tiempo que proporciona a las empresas en el equipo que lo utilicen en lugar

de crear un documento excel desde cero y programar cada celda tienen una aplicación que lo hace “solo”.

Si al ahorro de tiempo le sumamos la forma de verlo, es un proyecto muy cargado de la parte de UX y front ya que si haces que la experiencia de usuario sea única todo el mercado querrá la app, se puede explotar mucho por esa parte. Existe el reto de hacer de un algoritmo difícil, una interfaz “para niños”, lo ideal sin manual.

Además al tratarse de una aplicación en angularJS cargada del lado del cliente es muy ágil y la velocidad que va a tener a nivel de uso va a ser satisfactoria para el cliente. Al no tener base de datos reduce el tiempo de carga y de espera por queries.

La solicitud es una gran oportunidad que está destinada a todos aquellos que en su día a día tomen decisiones y hagan estudios comparativos, tienen resuelto el problema si quieren probar el modelo nuevo para dejar a un lado el modelo tradicional.

4.3 Casos de uso

4.3.1 Alcance

Los casos de uso describen de forma detallada las acciones que puede realizar el usuario en el sistema, teniendo en cuenta todos los caso así como las precondiciones y postcondiciones da cado uno de los casos identificados expandidos.

En primer lugar se hace una descripción de los actores que intervienen en el sistema, a continuación los casos de uso y posteriormente los casos de uso expandidos que se diferencian de los anteriores en que estos posen los escenarios y cada una de las acciones que se daría por caso de uso.

En medio de estos dos apartados, adjunto el diagrama de casos de uso en notación UML en el apartado denominado, diagrama de casos de uso. Con esto me permite

hacer una visión general de todas las funcionalidades para luego al entrar en cada uno ya tengamos una foto de lo general.

Antes de entrar en cada uno de ellos en el apartado de especificación de casos de uso incluyo una plantilla de la tabla donde describo cada uno para tener claro los campos que tengo en cuenta.

4.3.2 Actores del sistema

En primer lugar voy a especificar los actores del sistema que he identificado en el proyecto. Solo existe un actor que es el usuario de la aplicación web desarrollada, debido a que o es una app con login no hay diferentes perfiles sino uno único que hace uso del proyecto.

Este usuario va a ser cambiante ya que como he dicho en los stakeholders no es un público en concreto al que sólo se le puede aplicar, podrá ser desde una empresa que la compran para hacer uso de ella y tomar decisiones en el equipo de recursos humanos, así como en el equipo de desarrollo para elegir la tecnología con la que desarrollan (si tratásemos de una empresa tecnológica); hasta un particular para decisiones diarias de a qué colegio llevar a sus hijos, el coche qué comprarse o la casa ideal para su vida.

4.3.3 Diagrama de uso

Ilustración 9: Diagrama de uso

4.3.4 Especificaciones de casos de uso

Para especificar el comportamiento del sistema y la secuencia de interacciones entre este y el actor voy a utilizar casos de uso.

Tras realizar el análisis y describir cada uno de los requisitos funcionales y no funcionales del proyecto puede generarse los casos de uso de la aplicación.

Como he mencionado en el apartado de alcance de este mismo capítulo a continuación incluyo una "tabla plantilla" en la que describo antes de comenzar cada uno de los campos que va a tener cada caso de uso.

ID
Nombre
Requisitos relacionados
Objetivo
Descripción
Precondición
Postcondición

Tabla 8: Plantilla de casos de uso

- **ID**: identificador único de cada uno de los casos de uso.
    - CU_XX
        - CU: iniciales de caso de uso, (parte del identificador que se repite en todos los caso de uso).
        - XX: número de caso de uso, (parte del identificador que hace inequívoco a cada uno de ellos)
- **Nombre**: título que recibe el caso de uso.
- **Requisitos relacionados**: referencia a los requisitos de la sección anterior que se ven involucrados en el caso de uso en concreto.
- **Objetivo**: define el propósito del caso de uso.
- **Descripción**: breve descripción de la funcionalidad.

- **Precondición:** condiciones que deben de darse antes de poder realizar este caso de uso.
- **Postcondición:** condiciones en las que se queda el sistema una vez se ha terminado de realizar el caso de uso.

*En la tabla de casos de uso es imprescindible el campo de actores, donde se citan los actores que influyen en cada caso. Como esta aplicación solo se ha identificado un actor, el usuario de la app, por eso omito el campo ya que en todos es el mismo.

ID	CU_01
Nombre	Crear nodo
Requisitos relacionados	RF_01, RF_02, RF_14, RF_15
Objetivo	Crear un criterio que va a influir en tu decisión
Descripción	El usuario creará tantos nodos como criterios afecten a la elección. Es el primer paso de la app, lo que forma el árbol de decisión.
Precondición	-
Postcondición	Un nodo más en el árbol de decisión. Se habilitan las opciones de ordenar, editar o eliminar nodo y añadir nodo hijo.

Tabla 9: CU_01

ID	CU_02
Nombre	Editar nodo
Requisitos relacionados	RF_03, RF_06, RF_14, RF_15
Objetivo	Modificar los atributos de un nodo.
Descripción	El usuario puede modificar en cualquier momento los atributos de un nodo, estos son el nombre y los pesos de comparación por pares de sus nodos hijos.
Precondición	Haber creado un nodo para editar el nombre. Para poder editar nombre y pesos por pares de los nodos hijos, se deben haber creado como mínimo dos nodos hijos o más.

Postcondición	Atributos del nodo modificados.

Tabla 10: CU_02

ID	CU_03
Nombre	Eliminar nodo
Requisitos relacionados	RF_05, RF_14
Objetivo	Eliminar un criterio para la elección.
Descripción	Eliminar un nodo del árbol que ya no consideras o no quieres que influya en tu decisión final.
Precondición	Haber creado uno o más nodos.
Postcondición	Un nodo menos en el árbol.

Tabla 11: CU_03

ID	CU_04
Nombre	Ordenar nodo
Requisitos relacionados	RF_04, RF_14
Objetivo	Poder ordenar los nodos en cualquier momento.
Descripción	Ordenar todos los nodos en su lugar una vez creados todos. Ya que si el usuario crea los nodos desde el mismo nivel, ninguno será padre de otro, por lo que siempre da la posibilidad de reordenar.
Precondición	Haber creado dos o más nodos.
Postcondición	Árbol ordenado adecuadamente.

Tabla 12: CU_04

ID	CU_05
Nombre	Añadir nodo hijo
Requisitos relacionados	RF_07, RF_15
Objetivo	Añadir un nodo dentro de otro.
Descripción	Permite añadir un nodo en un nivel inferior dentro de otro ya creado para ir estableciendo una jerarquía de criterios en la decisión.

Precondición	Haber creado un nodo.
Postcondición	Un nodo hijo dentro del padre, que te activa la opción de colapsar los niveles desde el padre y si existen dos hijos de editar pesos por pares.

Tabla 13: CU_05

ID	CU_06
Nombre	Establecer un peso a sus nodos hijos.
Requisitos relacionados	RF_06, RF_20
Objetivo	Crear la matriz por pares del algoritmo AHP.
Descripción	Al editar un nodo cuando tiene más de dos hijos, el usuario puede establecer un peso de importancia por pares de cada uno de los hijos para obtener la matriz por pares y sacar la mejor decisión.
Precondición	Crear dos hijos de un mismo nodo padre.
Postcondición	Matriz por pares para hallar el resultado del algoritmo multicriterio.

Tabla 14: CU_06

ID	CU_07
Nombre	Introducir los candidatos para la decisión.
Requisitos relacionados	RF_08, RF_17, RF_18
Objetivo	Introducir cada una de las opciones a elección.
Descripción	El usuario debe introducir cada una de las alternativas para poderlas ponderar con la matriz por pares y el vector de ponderación, y sacar la opción que mejor cumple todos los criterios.
Precondición	Tener listo el árbol de decisión.
Postcondición	Una vez introducidos todos los candidatos el usuario podrá darle a completado para obtener los resultados gráficamente.

Tabla 15: CU_07

ID	CU_08
Nombre	Editar candidato
Requisitos relacionados	RF_09, RF_17, RF_18
Objetivo	Actualizar los atributos de un candidato.
Descripción	Editar los campos del candidato en todo momento una vez creado ante cualquier cambio.
Precondición	Haber creado uno o más candidatos.
Postcondición	Información del candidato modificada con los nuevos datos introducidos en el formulario.

Tabla 16: CU:08

ID	CU_09
Nombre	Eliminar candidato
Requisitos relacionados	RF_10
Objetivo	Permitir al usuario eliminar un candidato.
Descripción	Una vez creado el candidato puede que el usuario descarte o que no influya para su decisión, en este caso, puede borrar el candidato.
Precondición	Haber creado un candidato.
Postcondición	Candidato eliminado de la lista de opciones a la elección.

Tabla 17: CU_09

ID	CU_10
Nombre	Importar proyecto
Requisitos relacionados	RF_12, RF_14
Objetivo	Permitir al usuario importar un proyecto ya creado para partir de una base hecha.
Descripción	Como puede que siempre el usuario utilice el árbol con diferentes candidatos, o para retomarlo, le permite añadir un proyecto ya realizado.
Precondición	Haber exportado un proyecto o generar un archivo JSON para importar.
Postcondición	El dashboard será cargado con los datos del JSON importado.

Tabla 18: CU_10

ID	CU_11
Nombre	Exportar proyecto
Requisitos relacionados	RF_11
Objetivo	Permitir al usuario exportar el proyecto para guardarlo y reutilizarlo en un futuro.
Descripción	Exportar un proyecto para utilizarlo en todas tus decisiones con ese árbol y evitar rehacerlo cada vez que se abre la aplicación web.
Precondición	Tener mínimo un nodo creado en el árbol de decisión.
Postcondición	Archivo JSON guardado en la carpeta de descargas del usuario.

Tabla 19: CU_11

ID	CU_12
Nombre	Eliminar proyecto
Requisitos relacionados	RF_13
Objetivo	Permitir al usuario desechar su proyecto.
Descripción	Elimina un proyecto en cualquier momento con una confirmación previa y empieza desde cero.
Precondición	Haber creado mínimo un nodo del árbol.
Postcondición	El sistema se inicia y empieza la aplicación desde cero sin ningún dato ni de criterios ni candidatos.

Tabla 20: CU_12

ID	CU_13
Nombre	Obtener resultados
Requisitos relacionados	RF_19, RF_20
Objetivo	Mostrar los resultados del algoritmo AHP.
Descripción	Obtener los resultados de la mejor decisión, mostrando las matrices de comparación por pares y los candidatos ordenados.
Precondición	Tener el árbol de decisión y todos los candidatos creados.

Postcondición	Resultados del algoritmo, muestra a mejor elección en función de los criterios establecidos por el usuario.

Tabla 21: CU_13

4.3.5 Casos de uso en formato expandido

Una vez llegado a este punto voy a terminar detallar cada funcionalidad de la aplicación con los casos de uso en formato expandido.

Al igual que en el apartado de casos de uso se utiliza la misma platilla pero con una pequeña alteración, se incluye la acción del usuario y la respuesta del sistema en cada caso en orden y detalladamente.

ID	
Nombre	
Requisitos relacionados	
Objetivo	
Descripción	
Precondición	
Postcondición	
Acciones del usuario	Respuestas del sistema

Tabla 22: Plantilla de sasos de uso expandido

- **ID**: identificador único de cada uno de los casos de uso.
  - CUX_XX
    - CUX: iniciales de caso de uso, (parte del identificador que se repite en todos los caso de uso).
    - XX: número de caso de uso, (parte del identificador que hace inequívoco a cada uno de ellos)
- **Nombre**: título que recibe el caso de uso.
- **Requisitos relacionados**: referencia a los requisitos de la sección anterior que se ven involucrados en el caso de uso en concreto.

- **Objetivo**: define el propósito del caso de uso.
- **Descripción**: breve descripción de la funcionalidad.
- **Acción del actor**: consecución de acciones que lleva a cabo el actor para realizar la funcionalidad de cada caso de uso.
- **Respuesta del sistema**: respuestas que va dando el sistema a cada una de las acciones del usuario.

ID	CUX_01
Nombre	Crear nodo
Requisitos relacionados	RF_01, RF_02, RF_14, RF_15
Objetivo	Crear un criterio que va a influir en tu decisión
Descripción	El usuario creará tantos nodos como criterios afecten a la elección. Es el primer paso de la app, lo que forma el árbol de decisión.
Precondición	-
Postcondición	Un nodo más en el árbol de decisión. Se habilitan las opciones de ordenar, editar o eliminar nodo y añadir nodo hijo.
Acciones del usuario	Respuestas del sistema
1.El usuario introduce la URL de la app en el navegador.	
	2. Se muestra la app en el navegador
3.El usuario hace clic sobre el botón de crear nodo.	
	4. El sistema devuelve en la interfaz un modal con un formulario
5.El usuario debe rellenar el formulario con los datos del criterio.	
6.El usuario realiza el "submit" en el modal de creación.	
	7. El sistema almacena el nuevo criterio con los atributos en un JSON.
	8. La app muestra el nuevo nodo en el árbol de decisión para que lo vea el usuario creado.

Tabla 23: CUX_01

ID	CUX_02	
Nombre	Editar nodo	
Requisitos relacionados	RF_03, RF_06, RF_14, RF_15	
Objetivo	Modificar los atributos de un nodo.	
Descripción	El usuario puede modificar en cualquier momento los atributos de un nodo, estos son el nombre y los pesos de comparación por pares de sus nodos hijos.	
Precondición	Haber creado un nodo para editar el nombre. Para poder editar nombre y pesos por pares de los nodos hijos, se deben haber creado como mínimo dos nodos hijos o más.	
Postcondición	Atributos del nodo modificados.	
Acciones del usuario		Respuestas del sistema
1.Hacer clic sobre el botón de editar de un nodo creado.		
		2.El sistema devuelve un modal con los datos del formulario rellenados con los valores correspondientes.
3.El usuario cambia los parámetros que desee del criterio.		
4.El usuario hace submit del formulario.		
		5.El sistema actualiza la información del nodo en la aplicación
		6.El sistema refleja en el árbol el nuevo nodo actualizado.

Tabla 24: CUX_02

ID	CUX_03
Nombre	Eliminar nodo
Requisitos relacionados	RF_05, RF_14
Objetivo	Eliminar un criterio para la elección.
Descripción	Eliminar un nodo del árbol que ya no consideras o no quieres que influya en tu decisión final.
Precondición	Haber creado uno o más nodos.

Postcondición	Un nodo menos en el árbol.
Acciones del usuario	Respuestas del sistema
1.El usuario hace clic sobre el botón de eliminar del nodo que desee realizar la operación.	
	2.El sistema eliminará el nodo seleccionado y si tiene hijos elimina todos los hijos.
	3.La app pinta el árbol de decisión sin los criterios eliminados.

Tabla 25: CUX_03

ID	CUX_04
Nombre	Ordenar nodo
Requisitos relacionados	RF_04, RF_14
Objetivo	Poder ordenar los nodos en cualquier momento.
Descripción	Ordenar todos los nodos en su lugar una vez creados todos. Ya que si el usuario crea los nodos desde el mismo nivel, ninguno será padre de otro, por lo que siempre da la posibilidad de reordenar.
Precondición	Haber creado dos o más nodos.
Postcondición	Árbol ordenado adecuadamente.
Acciones del usuario	Respuestas del sistema
1.El usuario con un “drap and drog” puede ordenar un nodo en cualquier momento de una forma sencilla o agruparlo.	
	2. El sistema muestra en tiempo real que ubicación tomaría el nodo en cada momento con líneas discontinuas para que el usuario lo perciba mejor.

Tabla 26: CUX_04

ID	CUX_05
Nombre	Añadir nodo hijo

Requisitos relacionados	RF_07, RF_15
Objetivo	Añadir un nodo dentro de otro.
Descripción	Permite añadir un nodo en un nivel inferior dentro de otro ya creado para ir estableciendo una jerarquía de criterios en la decisión.
Precondición	Haber creado un nodo.
Postcondición	Un nodo hijo dentro del padre, que te activa la opción de colapsar los niveles desde el padre y si existen dos hijos de editar pesos por pares.
Acciones del usuario	Respuestas del sistema
1.El usuario hace clic sobre el botón de crear nodo dentro del nodo ya existente que quiere que ejerza de padre.	
	2. El sistema devuelve en la interfaz un modal con un formulario
3.El usuario debe rellenar el formulario con los datos del criterio.	
4.El usuario realiza el "submit" en el modal de creación.	
	5. El sistema almacena el nuevo criterio con los atributos en un JSON.
	6.La app muestra el nuevo nodo en el árbol de decisión para que lo vea el usuario creado.

Tabla 27: CUX_05

ID	CUX_06
Nombre	Establecer un peso a sus nodos hijos.
Requisitos relacionados	RF_06, RF_20
Objetivo	Crear la matriz por pares del algoritmo AHP.
Descripción	Al editar un nodo cuando tiene más de dos hijos, el usuario puede establecer un peso de importancia por pares de cada uno de los hijos para obtener la matriz por pares y sacar la mejor decisión.
Precondición	Crear dos hijos de un mismo nodo padre.
Postcondición	Matriz por pares para hallar el resultado del algoritmo multicriterio.

Acciones del usuario	Respuestas del sistema
1.Hacer clic sobre el botón de editar de un nodo creado que posea al menos dos hijos.	
	2.El sistema devuelve un modal con los datos del formulario rellenados con los valores correspondientes.
3.El usuario establece con un slider los pesos comparativos por pares de los criterios hijo.	
4.El usuario hace submit del formulario.	
	5.El sistema actualiza la información del nodo en la aplicación

Tabla 28: CUX_06

ID	CUX_07
Nombre	Introducir los candidatos para la decisión.
Requisitos relacionados	RF_08, RF_17, RF_18
Objetivo	Introducir cada una de las opciones a elección.
Descripción	El usuario debe introducir cada una de las alternativas para poderlas ponderar con la matriz por pares y el vector de ponderación, y sacar la opción que mejor cumple todos los criterios.
Precondición	Tener listo el árbol de decisión.
Postcondición	Una vez introducidos todos los candidatos el usuario podrá darle a completado para obtener los resultados gráficamente.
Acciones del usuario	Respuestas del sistema
1.El usuario hace clic sobre el menú lateral izquierdo abajo para añadir un nuevo candidato a la elección.	
	2.La aplicación devuelve un modal que sale de la parte derecha lateral con el formulario de los candidatos.
	3.El sistema rellena las labels de los inputs del formulario con cada uno de los criterios creados por el usuario.

4.El usuario rellena el formulario con los datos del candidato.	
5.El usuario hace submit del formulario.	
	6.El sistema almacena un nuevo candidato en el array de objetos candidatos.
	7.El sistema muestra mediante la interfaz en el menú lateral izquierdo el nuevo candidato para que lo vea el usuario ya creado.

Tabla 29: CUX_07

ID	CUX_08
Nombre	Editar candidato
Requisitos relacionados	RF_09,RF_17,RF_18
Objetivo	Actualizar los atributos de un candidato.
Descripción	Editar los campos del candidato en todo momento una vez creado ante cualquier cambio.
Precondición	Haber creado uno o más candidatos.
Postcondición	Información del candidato modificada con los nuevos datos introducidos en el formulario.
Acciones del usuario	Respuestas del sistema
1.El usuario hace clic sobre el botón de editar de uno de los candidatos que desee modificar.	
	2.El sistema abre un formulario en la parte lateral derecha.
	3.El sistema rellena los campos del candidato en el formulario.
4.El usuario modifica los atributos que quiera cambiar del candidato.	
5.El usuario hace submit del formulario para actualizar.	
	6.El sistema actualiza los nuevos datos en el array candidatos.
	7.El sistema pinta de nuevo el candidato en el menú lateral izquierdo para la

	visualización del usuario.

Tabla 30: CUX_08

ID	CUX_09
Nombre	Eliminar candidato
Requisitos relacionados	RF_10
Objetivo	Permitir al usuario eliminar un candidato.
Descripción	Una vez creado el candidato puede que el usuario descarte o que no influya para su decisión, en este caso, puede borrar el candidato.
Precondición	Haber creado un candidato.
Postcondición	Candidato eliminado de la lista de opciones a la elección.
Acciones del usuario	Respuestas del sistema
1.El usuario hace clic sobre el botón de eliminar de uno de los candidatos que desee eliminar.	
	2.El sistema elimina el candidato seleccionado y actualiza los datos en el array candidatos.
	3.El sistema elimina de la interfaz el candidato para que se vea que ya no forma parte de las alternativas a la decisión del usuario.

Tabla 31: CUX_09

ID	CUX_10
Nombre	Importar proyecto
Requisitos relacionados	RF_12, RF_14
Objetivo	Permitir al usuario importar un proyecto ya creado para partir de una base hecha.
Descripción	Como puede que siempre el usuario utilice el árbol con diferentes candidatos, o para retomarlo, le permite añadir un proyecto ya realizado.
Precondición	Haber exportado un proyecto o generar un archivo JSON para importar.
Postcondición	El dashboard será cargado con los datos del JSON

importado.	
Acciones del usuario	Respuestas del sistema
1.El usuario abre la aplicación con la URL correspondiente.	
	2.El sistema devuelve la app.
3.El usuario hace clic sobre el botón de importar del navbar, localizado en la parte superior de la aplicación.	
	4.El sistema devuelve un modal con un campo para introducir el directorio del archivo en JSON.
5.El usuario selecciona el archivo que quiere importar a la aplicación web.	
	6.El sistema lee el archivo adjunto.
	7. La app pinta el árbol de decisión que ha importado el usuario en su archivo JSON adjunto.

Tabla 32: CUX_10

ID	CUX_11
Nombre	Exportar proyecto
Requisitos relacionados	RF_11
Objetivo	Permitir al usuario exportar el proyecto para guardarlo y reutilizarlo en un futuro.
Descripción	Exportar un proyecto para utilizarlo en todas tus decisiones con ese árbol y evitar rehacerlo cada vez que se abre la aplicación web.
Precondición	Tener mínimo un nodo creado en el árbol de decisión.
Postcondición	Archivo JSON guardado en la carpeta de descargas del usuario.
Acciones del usuario	Respuestas del sistema
1.El usuario hace clic sobre el botón de exportar de la barra superior de la app.	
	2. El sistema muestro un modal de confirmación de la operación.
3.El usuario hace clic sobre cancelar.	
	4.El sistema cancela la operación y no se

	exporta ningún archivo JSON del árbol.
3.El usuario hace clic sobre aceptar exportación.	
	4.El sistema devuelve un archivo JSON con el árbol para que el usuario guarde el proyecto y pueda retomarlo o reutilizarlo en cualquier momento.

Tabla 33: CUX_11

ID	CUX_12
Nombre	Eliminar proyecto
Requisitos relacionados	RF_13
Objetivo	Permitir al usuario desechar su proyecto.
Descripción	Elimina un proyecto en cualquier momento con una confirmación previa y empieza desde cero.
Precondición	Haber creado mínimo un nodo del árbol.
Postcondición	El sistema se inicia y empieza la aplicación desde cero sin ningún dato ni de criterios ni candidatos.
Acciones del usuario	Respuestas del sistema
1.El usuario hace clic sobre la barra superior de navbar el botón de eliminar proyecto.	
	2. El sistema muestra un modal de confirmación al usuario de la acción seleccionada.
3.El usuario cancela la operación.	
	4.El sistema se mantiene y no lleva a cabo la acción seleccionada.
3.El usuario presiona sobre eliminar	
	4.La aplicación elimina el proyecto, ya son los criterios como los candidatos introducidos.
	5.El sistema se resetea todos los valores empiezan desde cero y queda el proyecto completamente eliminado en la interfaz,

	no hay árbol de decisión ni candidatos en caso de que existieran.

Tabla 34: CUX_12

ID	CUX_13
Nombre	Obtener resultados
Requisitos relacionados	RF_19, RF_20
Objetivo	Mostrar los resultados del algoritmo AHP.
Descripción	Obtener los resultados de la mejor decisión, mostrando las matrices de comparación por pares y los candidatos ordenados.
Precondición	Tener el árbol de decisión y todos los candidatos creados.
Postcondición	Resultados del algoritmo, muestra a mejor elección en función de los criterios establecidos por el usuario.
Acciones del usuario	Respuestas del sistema
1.El usuario hace clic sobre la barra superior de navbar el botón de siguiente.	
	2. El sistema muestra un modal de confirmación al usuario de la acción seleccionada.
3.El usuario cancela la operación.	
	4.El sistema se mantiene y no lleva a cabo la acción seleccionada.
3.El usuario presiona sobre continuar.	
	4.El sistema calcula el algoritmo AHP que ha sido implementado con los pesos y criterios establecidos por el usuario en el paso anterior.
	5.La aplicación muestra una nueva pantalla con los candidatos ordenados en orden de decisión y las matrices creadas de comparación.

Tabla 35: CUX_13

4.4 Requisitos del sistema

4.4.1 Definición

Para definir los diferentes aspectos que debe cumplir la aplicación a desarrollar uso los requisitos de software, sirviendo con estos, de guía a la hora de desarrollar. A su vez permite de una manera ordenada recoger todo lo pedido por el cliente y tenerlo documentado de alguna manera para que a la finalización del proyecto se pueda comprobar si se han cumplido los diferentes objetivos que se propusieron en la fase de análisis.

En las secciones siguientes voy a proceder a detallar de forma exhaustiva los requisitos para la implementación de una aplicación web con el algoritmo multicriterio basado en AHP, diferenciando dos grandes grupos, para facilitar la comprensión de los requerimientos del sistema:

- **Requisitos funcionales:** son los que definen las funcionalidades que ofrece la aplicación al usuario.
  - **Requisitos de usuario**: determinan la funcionalidad que debe hacer el usuario para cada acción.
  - **Requisitos de funcionalidad:** determinan el funcionamiento del sistema propiamente dicho.

- **Requisitos no funcionales:** definen el resto de aspectos del sistema, es decir, usabilidad de la aplicación, interfaz, mantenimiento, calidad, seguridad, etc.
  - **Requisitos de seguridad:** determinan lo necesario para crear una aplicación segura.
  - **Requisitos de disponibilidad:** fijan lo necesario para tener disponibilidad completa de la herramienta.
  - **Requisitos de mantenibilidad:** establecen el mantenimiento requerido para dar soporte a la interfaz.

- **Requisitos de interfaz:** marcan cómo debe ser la línea de la interfaz de una manera general pero imprescindible.

Para la definición de todos los requisitos emplearé una tabla plantilla que adjunto a continuación con una breve descripción de cada campo.

ID
Nombre
Tipo
Fuente
Prioridad
Descripción

Tabla 36: Plantilla de requisitos software

- **ID:** identificador único de cada uno de los requisitos de software.
  - RY_XX
    - R: inicial de requisito, (parte del identificador que se repite en todos los caso de uso).
    - Y: puede tomar dos valores:
      - F: para los requisitos funcionales.
      - NF: para requisitos no funcionales.

XX: número de caso de uso, (parte del identificador que hace inequívoco a cada uno de ellos)

Nombre: título que recibe el requisito.

Fuente: referencia de quien es el que ha pedido dicho requisito.

Prioridad: define la trascendencia del requisito. Puede tomar un valor entre estos tres:

Alta: La importancia de este requisito es máxima, de tal manera que su desarrollo será prioritario frente a otros de categoría inferior.

Media: Su importancia es parcialmente destacable y por tanto han de ser satisfechos con carácter mandatorio, si bien con una relevancia menor que los de prioridad alta.

Baja: Este requisito tiene una relevancia reducida y por tanto se situará en la cola a la hora de implementar.

Descripción: breve descripción de la funcionalidad.

4.4.2 Requisitos funcionales

Requisitos de usuario

ID	RF_01
Nombre	Crear un criterio
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario deberá poder crear un nodo para la elaboración de un árbol de decisión, poniendo a su disposición un botón en la dashboard de la app web.

Tabla 37: RF_01

ID	RF_02
Nombre	Rellenar datos de un criterio
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario al hacer clic sobre el botón de crear nodo, recibirá un modal con un formulario con un input de nombre para asignar al criterio.

Tabla 38: RF_02

ID	RF_03
Nombre	Modificar un criterio
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario debe poder editar un criterio y modificar sus datos en cualquier momento.

Tabla 39: RF_03

ID	RF_04
Nombre	Agrupar criterios
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario debe tener la posibilidad de agrupar criterios y ordenarlos en cada momento una vez creados.

Tabla 40: RF_04

ID	RF_05
Nombre	Eliminar criterio
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario debe tener la posibilidad de eliminar un criterio que haya creado.

Tabla 41: RF_05

ID	RF_06
Nombre	Establecer comparaciones por pares
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario puede establecer los pesos de importancia en comparativa par a par de cada uno de sus hijos

Tabla 42: RF_06

ID	RF_07
Nombre	Crear un nodo hijo
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario puede crear un nodo hijo de otro, ya que un criterio puede constar de otros a su vez.

Tabla 43: RF_07

ID	RF_08
Nombre	Crear candidatos
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario puede crear los candidatos que baraje para la decisión desde un menú.

Tabla 44: RF_08

ID	RF_09
Nombre	Editar candidatos
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario puede editar los candidatos que ya ha creado para la decisión.

Tabla 45: RF_09

ID	RF_10
Nombre	Eliminar candidatos
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario puede borrar los candidatos que ya ha creado para la decisión.

Tabla 46: RF_10

ID	RF_11
Nombre	Guardar un proyecto
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario puede guardar en cualquier estado el proyecto para retomarlo en el momento que desee.

Tabla 47: RF_11

ID	RF_12
Nombre	Importar un proyecto
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario puede importar en cualquier estado el proyecto para retomarlo como lo guardo.

Tabla 48: RF_12

ID	RF_13
Nombre	Eliminar un proyecto
Tipo	Usuario
Fuente	Cliente
Prioridad	Alta
Descripción	El usuario puede eliminar en cualquier estado el proyecto si ya no desea mantenerlo.

Tabla 49: RF_13

Requisitos de funcionalidad

ID	RF_14
Nombre	La aplicación muestra el árbol de decisión de manera visual
Tipo	Funcionalidad
Fuente	Cliente
Prioridad	Media
Descripción	La aplicación muestra el árbol de decisión que crea el usuario con los criterios de manera visual.

Tabla 50: RF_14

ID	RF_15
Nombre	La aplicación muestra el nombre del criterio.
Tipo	Funcionalidad
Fuente	Analista
Prioridad	Media
Descripción	La aplicación muestra el nombre de criterio para identificarlo en el nodo

	del árbol.

Tabla 51: RF_15

ID	RF_16
Nombre	La aplicación muestra el nombre del candidato.
Tipo	Funcionalidad
Fuente	Analista
Prioridad	Alta
Descripción	La aplicación muestra el nombre del candidato para identificarlo en el menú lateral de “mis candidatos”.

Tabla 52: RF_16

ID	RF_17
Nombre	Listar candidatos
Tipo	Funcionalidad
Fuente	Analista
Prioridad	Alta
Descripción	El usuario puede ver en un menú siempre todos los candidatos que ha introducido para su elección.

Tabla 53: RF_17

ID	RF_18
Nombre	Establecer los atributos de los candidatos
Tipo	Funcionalidad
Fuente	Analista
Prioridad	Media
Descripción	El usuario puede rellenar los criterios hijos que ha establecido en árbol como atributos de cada uno de los candidatos.

Tabla 54: RF_18

ID	RF_19
Nombre	Visualización de resultados
Tipo	Funcionalidad
Fuente	Analista
Prioridad	Media

Descripción	El usuario visualiza los datos de los candidatos a la elección una vez aplicado los criterios con los resultados.

Tabla 55: RF_19

ID	RF_20
Nombre	Utilización del algoritmo multicriterio AHP
Tipo	Funcionalidad
Fuente	Analista
Prioridad	Media
Descripción	La aplicación mostrará las estadísticas de los resultados con el árbol creado bajo el algoritmo implementado AHP.

Tabla 56: RF_20

4.4.3 Requisitos no funcionales

Requisitos de seguridad

ID	RNF_01
Nombre	Tratamiento de datos
Tipo	Requisito de seguridad
Fuente	Analista
Prioridad	Alta
Descripción	El sistema deberá tratar la información de acuerdo a la Ley Orgánica 15/1999 de protección de Datos de Carácter Personal

Tabla 57: RNF_01

Requisitos de disponibilidad

ID	RNF_02
Nombre	Disponibilidad total
Tipo	Requisito de disponibilidad
Fuente	Cliente

Prioridad	Alta
Descripción	La aplicación debe estar disponible las 24 horas del día durante los 7 días de la semana.

Tabla 58: RNF_02

ID	RNF_03
Nombre	Conexión a internet
Tipo	Requisito de disponibilidad
Fuente	Analista
Prioridad	Alta
Descripción	La aplicación para su funcionamiento tiene que tener conexión a internet ya sea por wifi o por red.

Tabla 59: RNF_03

ID	RNF_04
Nombre	Sistema operativo
Tipo	Requisito de disponibilidad
Fuente	Cliente
Prioridad	Alta
Descripción	Se podrá acceder a la aplicación desde navegadores en dispositivos Windows, Linux y Mac.

Tabla 60: RNF_04

ID	RNF_05
Nombre	Navegador
Tipo	Requisito de disponibilidad
Fuente	Cliente
Prioridad	Alta
Descripción	La aplicación deberá funcionar en los navegadores google chrome, firefox y safari.

Tabla 61: RNF_05

Requisitos de mantenibilidad

ID	RNF_06
Nombre	Mantenimiento de la app
Tipo	Requisito de mantenibilidad
Fuente	Cliente
Prioridad	Media
Descripción	La aplicación deberá tener un mantenimiento periódico cada 6 meses e ir cubriendo los bugs o posibles funcionalidades nuevas.

Tabla 62: RNF_06

Requisitos de interfaz

ID	RNF_07
Nombre	Facilidad al usuario
Tipo	Requisito de interfaz
Fuente	Cliente
Prioridad	Media
Descripción	La aplicación deberá mostrar en una misma pantalla las funcionalidades de una manera simple y debe mostrar consistencia en el diseño.

Tabla 63: RNF_07

ID	RNF_08
Nombre	Visualización de la interfaz
Tipo	Requisito de interfaz
Fuente	Cliente
Prioridad	Alta
Descripción	La aplicación deberá tener un aspecto sencillo y está diseñada para que cualquier usuario pueda utilizarla.

Tabla 64: RNF_08

ID	RNF_09
Nombre	Idioma de la interfaz
Tipo	Requisito de interfaz

Fuente	Cliente
Prioridad	Alta
Descripción	La aplicación deberá estar en español.

Tabla 65: RNF_09

Requisitos de sistema

ID	RNF_10
Nombre	Librerías de interfaz
Tipo	Requisito de sistema
Fuente	Desarrollador
Prioridad	Media
Descripción	Para desarrollar la aplicación se utilizarán las librerías: angular-ui-tree, angular-material y bootstrap.

Tabla 66: RNF_10

ID	RNF_11
Nombre	Lenguaje de desarrollo del lado del cliente
Tipo	Requisito de sistema
Fuente	Desarrollador
Prioridad	Media
Descripción	El código del lado del cliente será desarrollado en angularJS.

Tabla 67: RNF_11

4.5 Análisis de clases

En esta sección del documento a partir de los requisitos de software y casos de uso extraídos previamente voy a definir las clases necesarias ya metiéndome en un lado de implementación del código de la interfaz.

Hablamos de clases, ya que es el lenguaje estándar, con el diagrama de clases que vendrá adjunto al final de esta sección; pero al tratarse de una app web implementada con angularJS no voy a tratar con clases sino con componentes.

Cada caso de uso podrá ser implementado por uno o más componentes de tal forma que cada uno de los componentes cuenta con una directiva/controlador que será donde desarrolle las funciones que debe realizar, y por otro lado tendrá un HTML con la interfaz visual que ve el usuario con los estilos de la página .less.

4.5.1 Identificación de los componentes

Como en el apartado anterior he mencionado, la aplicación debe implementarse en angularJS por lo que tendré un componente por cada sección de la interfaz donde se implementan cada una de las funciones correspondientes.

A continuación se citan cada una de las funciones acompañadas de una breve descripción para identificarlas y ver la funcionalidad que cubren:

- **Navbar component:**
  - **navbar.js:** archivo javaScript que contiene las funciones del header de la aplicación, es decir las acciones que tiene repercusión sobre el proyecto entero.
    - **importProyect:** completa el requisito de importar un proyecto guardado, esta función lee un archivo JSON que contiene un árbol de decisión anteriormente guardado y lo pinta en el dashboard, para pasarle los parámetros utilizo un servicio.
    - **exportProyect:** exporta un proyecto, es decir, guarda el proyecto en el que se está trabajando en la app web en ese momento para poder reutilizarlo, al pulsar sobre la acción te abre un modal en el que confirmas que quieres guardar el proyecto. Se descarga un archivo JSON de objetos nodos.
    - **continueProyect:** una vez finalizado la intrusión de datos necesarios para el algoritmo AHP multicriterio esta función nos manda un modal de confirmación y seguidamente será la encargada de implementar el algoritmo con las matrices por pares y los vectores de ponderación para sacar la solución.

- **deleteProyect:** borra los datos de ese proyecto, tanto de los candidatos como del árbol introducido, de esta forma es como crear un nuevo proyecto sin antecedentes, siempre y cuando el usuario acepte la confirmación que le saldrá.

- **navbar.html:** hoja html que contiene la cabecera de la interfaz, donde a la izquierda aparece el nombre de la app acompañado de un icono para volver a la home en caso de querer retroceder, y a la izquierda los botones representativos de cada una de las acciones anteriores.

- **navbar.less:** archivo css que da estilo y formato al header dónde tengo el color corporativo de la app y los iconos en una línea blanca buscando siempre los principio de diseño.

- **Dashboard component:**
  - **dashboard.js:** Archivo del componente de la pantalla principal que contiene todos los métodos y funciones para el desarrollo del árbol de decisión implementado con la librería de angular-ui-tree. Los métodos son lo siguientes:
    - **newItem:** función que añade un nuevo nodo al árbol de decisión llamando a un modal con un formulario, el cual compruebo su información, si es válida se añade al objeto JSON de los nodos del árbol y posteriormente con la instancia $uibModalInstance cierro el modal para mostrar en la pantalla el nuevo nodo.
    - **newEditNodeModal:** función responsable de la funcionalidad de edición de un nodo para modificar tanto el nombre del criterio si es necesario, o el peso de los hijos en la comparativa por pares si tiene dos o más hijos el padre.
    - **remove:** función que elimina un nodo del árbol de decisión y por lo tanto accede al archivo JSON elimina esa posición que ocupa en el array con toda su información.
    - **newSubItem:** añade un nodo hijo una vez creado un nodo, llama al mismo modal que si se crea un nodo desde cero, pero se le pasa como parámetro el nodo padre, ya que el hijo se debe introducir dentro de la misma posición en el array, de forma anidada.
    - **collapseAll:** cuando un nodo tiene hijos, es decir, es padre, tiene una funcionalidad más que el resto, puede ser colapsado para poder visualizar el resto de nodos de una ojeada si el árbol es largo.

- **expandAll:** cuando un nodo tiene hijos, es decir, es padre, tiene una funcionalidad más que el resto, que es la anterior; la que deshace dicha acción es esta, expandir para poder visualizar los hijos.
- **moveLastToTheBeginning:** método que me permite mover el nodo con la lógica del drap and drog en cualquier momento de una manera sencilla.

- **dashboard.html:** hoja html en la que se "pinta" el árbol de decisión según el usuario lo vaya creando, acompañado cada nodo de las acciones correspondientes de edición, borrado, añadir, ordenar y colapsar/expandir.

- **dashboard.less:** hoja de estilos css donde a cada clase utilizada en el html le doy los estilos que quiera para facilitar la experiencia de usuario y el la línea de diseño que quería perseguir de sencillez y limpieza.

- **Sidenav-left component:**
  - **sidenav-left.js:** archivo javaScript que implementa las funciones de los candidatos, todas las acciones que recaen sobre ellos. Son muy similares a las de los nodos del árbol pero con algunas modificaciones o restricciones.
    - **addCandidate:** método que implementa la función de añadir un nuevo candidato. Abre un formulario a la derecha que te crea un input por cada nodo hijo que exista en el árbol de decisión, el usuario debe rellenar cada uno de los campos y guardar. Al hacer el submit la aplicación crea un candidato lo almacena en un array de objetos candidatos y a continuación se lo muestra a la vista para que lo muestre al usuario.
    - **editCandidate:** función para editar los atributos de un candidato, al igual que la anterior abre el sidenav derecho pero ya con los campos rellenados y le permite al usuario cambiar el que desee y volver a guardar, la función una vez el usuario da al submit, se le pasa el id del candidato, lo busca en el array de candidatos y guarda de nuevo la información actualizado.
    - **removeCandidate:** método para eliminar candidatos, al hacer clic sobre el botón de eliminar se llama a esta función desde la vista al controlador pasandole como parametros el id del candidato, por lo que recorre el array de objetos candidatos y cuando el id sea igual que el id que ha recibido por parámetro lo elimina con la función splice(i,1).

- **sidenav-left.html:** archivo html situado al lado izquierdo de la pantalla contenedor de los candidatos a la elección, tiene tanto las acciones que se pueden realizar sobre los candidatos como la lista de candidatos introduces. En resumen, es una vista de gestión de los candidatos de una manera simple y limpia.

- **sidenav-left.less:** hoja de estilos css en la que le doy los estilos y efectos correspondientes a cada uno de los elementos mediante clases fijadas en el html, por ejemplo, el título de los sidenav siempre tendrá un tamaño y una línea inferior como cabecera.

- **Sidenav-right component:**
  - **sidenav-right.js:** archivo javaScript que básicamente va a tener dos métodos y el proceso de abrir o cerrar ya que se trata de un menú que contiene un formulario y a su vez dependiente de la interacción sobre un botón.
    - **openSidenavRight:** función encargada de abrir el menú lateral derecho cuando el botón de añadir candidato o el de editar es accionado. Le llega una señal desde el servicio que ha recibido del componente sidenav-left y abre el formulario para rellenarlo.
    - **infoCandidate:** cuando una de las acciones anteriores: añadir o editar candidatos tiene lugar, se llama a la función openSidenavRight y una vez abierto el formulario y rellenado al hacer el submit se mandan al otro controlador (sidenav-left) para que guarde la información del candidato con las funciones addCandidate y editCandidate anteriormente descritas.

  - **sidenav-right.html:** vista del menú lateral derecho que se muestra en función del menú lateral izquierdo si añade o edita un candidato. Contiene un formulario en el que se muestra los inputs, uno por cada nodo hijo del árbol, y se guarda la información en el array de candidatos correspondiente.

  - **sidenav-right.less:** estilos css que adquieren cada uno de los componentes de la plantilla html anterior que se les ha fijado una clase correspondiente y deben tener una tipografía o estilos correspondientes para mantener la consistencia y coherencia a lo largo de la aplicación.

- **index.html:** en angularJs es necesario tener una página html que es esta, la cual llama a las demás plantillas html y es la que reúne todo y forma la interfaz de la aplicación. Al arrancar la app se carga el index.html y es este quien tiene incrustado el código del resto.

4.5.2 Diagrama de clases

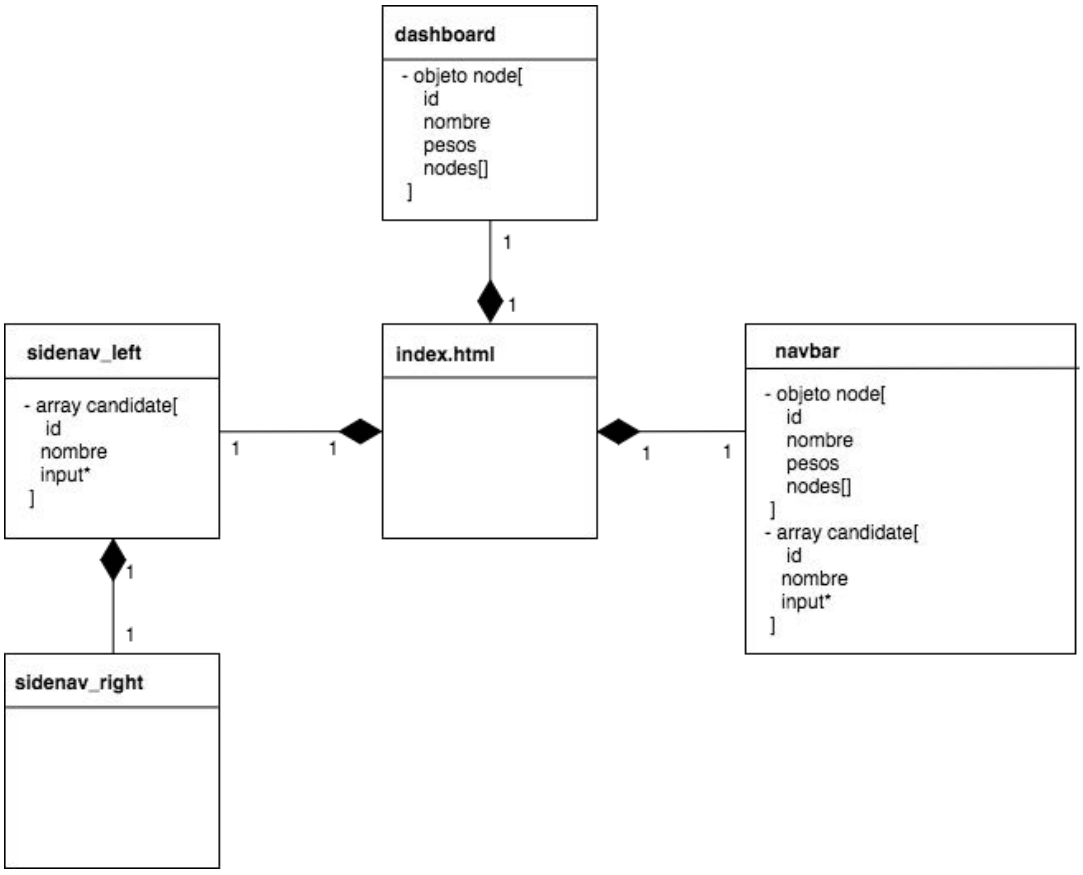


Ilustración 10: Diagrama de clases

El diagrama anterior es muy básico he incluido simplemente los atributos de cada una de las clases a nivel genérico ya que los métodos los he descrito en el apartado anterior para que no se sobrecargue en exceso el diagrama y al final sea más confuso que simple ver la arquitectura general del código.

La unión entre unas clases, en este caso componente, y otras la he establecido como agrupación, ya que no puede existir un navbar si no es llamado por el index, o no puede aparecer un dashboard si no es inyectado por el index.html, pero todo esto más que una clase hay que recordar siempre que no es un lenguaje orientado a objetos, sino que son componentes que necesitan de un index para aparecer en la interfaz.

5. Diseño

Una vez realizado el estudio de la propuesta de proyecto detalladamente analizando los casos de uso y los requisitos de software para cubrir todas las necesidades del cliente, entro en el apartado de diseño el cual lo voy a dividir en dos subapartados.

Por un lado expondré el diseño de la arquitectura del sistema, a nivel físico y lógico entrando al detalle de como se ha construido la aplicación web "Master Search".

Por otro lado, dedicaré la otra sección al diseño de la interfaz, parte que en este proyecto como ya he mencionado tiene gran importancia debido a la experiencia de usuario que se quiere alcanzar y que forma parte de requisito importante por parte de cliente.

5.1 Arquitectura del sistema

Como he comentado en el apartado de hardware decidí utilizar la tecnología de angularJS para implementar esta aplicación por las razones que se describen en dicha sección.

En la arquitectura del sistema actual no se incluye ningún tipo de base de datos ya que no es un requisito del cliente, es más, el cliente dijo que sería mejor sin base de datos porque al querer una aplicación general para muchos mercados no se podría tener en cuenta todo.

Si es verdad, que antes de entrar en detalle de la arquitectura que he implementado quiero dejar un breve y sintético esquema de cómo sería la app con una base de datos, ya que se ha dejado preparada para enganchar en cualquier momento y es una posible mejora del sistema para clientes específicos.

Explico un poco más claro, cuando acabo de decir para clientes específicos me refiero al personalizar la app web para un cliente, voy a poner un ejemplo para verlo más claro. Un concesionario adquiere la aplicación para el departamento comercial, a la hora de vender un coche a un cliente le hacen un estudio previo en función de sus necesidades y criterios que influyen en su decisión. En este caso se le puede ofrecer a la empresa personalizar la app conectándola con la base de datos y en esa BBDD tienen todos los coches de su concesionario, de esta forma no tienen que estar metiendo los candidatos continuamente.

En este caso de la customización de la app la arquitectura sería como en este diagrama, lo adjunto para que se vea que es sencillo y cabe la posibilidad aunque actualmente esté pensada y desarrollada sin base de datos.

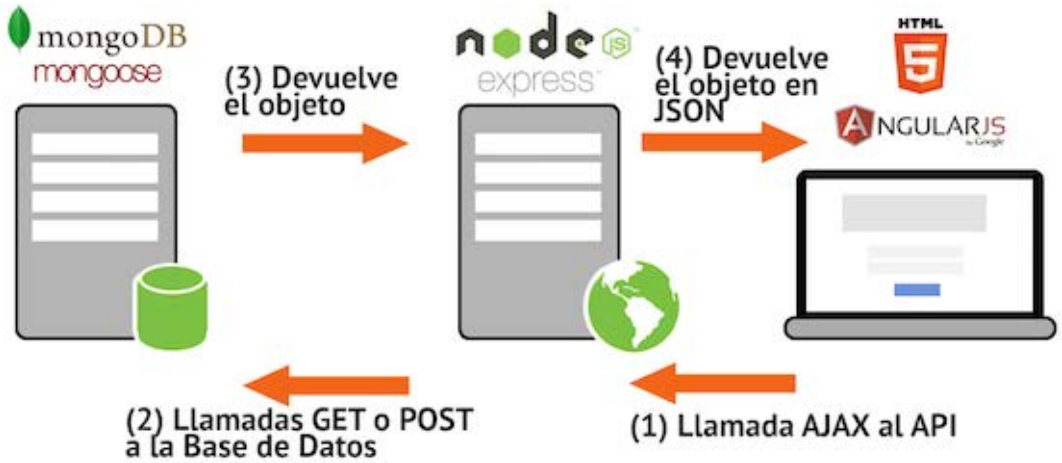


Ilustración 11: Arquitectura con BBDD

Ahora entramos a describir la arquitectura que he empleado para desarrollar una herramienta web para la resolución de problemas de decisión multicriterio con AHP y la tecnología de angujarJS.

En primer lugar adjunto un esquema que hace una descripción general de la arquitectura que emplea esta tecnología y a continuación describo cada uno de los componentes del esquema.

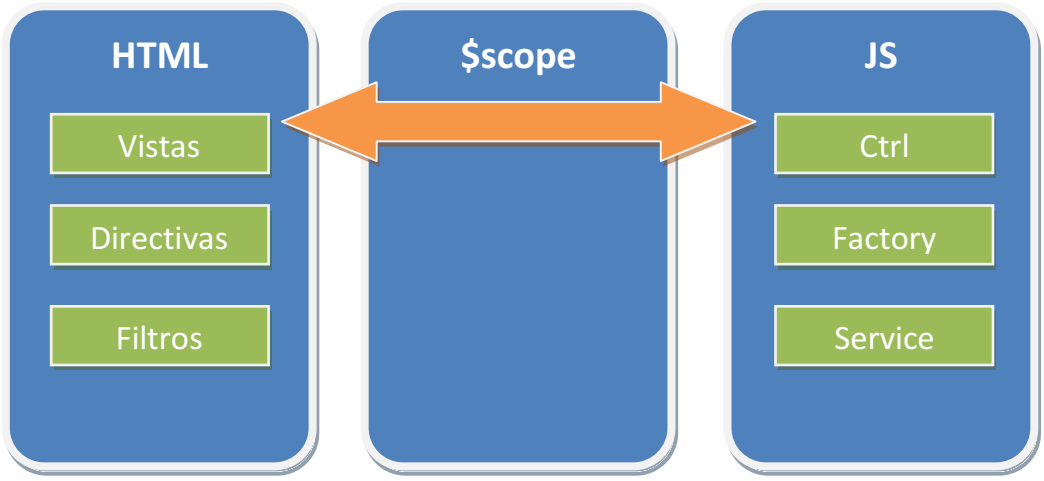


Ilustración 12: Lógica angular

Vistas

La base de la aplicación se construirá en el index.html, ya que durante tosa la aplicación el usuario se encuentra en él, encargado de inyectar el código del resto de HTML de cada uno de los componentes que contiene el código. Es decir, actúa a modo de frame, abriendo el resto de vistas dentro de unos de sus elementos del DOM.

Las partes a resaltar dentro de este archivo son las siguientes:

<html lang="es" ng-app="MasterSearch">

En esta línea se define con una directiva de angular “ng-app” que la aplicación de encuentra dentro he dicho elemento del DOM, en este caso, se ha colocado en el HTML, el elemento engloba todas las vistas de los componentes de la app.

En la línea superior del css debe incluirse en caso de que se incluya una librería de estilos para que afecte a todos los archivos css y desde ahí cojan los HTML dicho estilo, en este caso se incluyen la de angular-material y la de Bootstrap como se ha definido en el apartado de librerías utilizadas.

<link href=”directorioDeBootStrapYAngularMaterial” rel=”stylesheet”>

Posteriormente incluyo una línea de código que indica donde se encuentran los scripts de código, así como los básicos generales de la aplicación como es angular-route.js, para hacer funcionar el sistema modelo-vista-controlador.

<script src=”lib/angular/angular-route.js”></script>

<script src=”js/controllers.js”></script>

Por último y prácticamente lo más importante de la vista y en concreto del archivo index.html, es la línea que escribo a continuación.

***<div ng-view> ***

Esta simple línea es en la que se indica con una directiva de angularJS concretamente ng-view, que dentro de este elemento DOM es donde las vistas correspondientes a cada ruta se van a mostrar.

Directivas

Las directivas son el grueso dentro de esta tecnología. Mediante el uso de ellas se podrá extender de la sintaxis HTML y darle el comportamiento que desee. Se pueden crear directivas a nivel de elemento, de atributo, de clase o de comentario. Uno de los ejemplos en los que han sido utilizadas en la aplicación es a nivel de atributo para declarar una directiva que es hover y cada vez que estas encima de un elemento de la

lista de candidatos se incluye la directiva y hace el efecto correspondiente como en mi caso una animación de izquierda a derecha con un cambio de color de fondo.

Otra aplicación de estas seria para reemplazar etiquetas HTML por directivas de angular ya existentes que me facilitan la implementación en muchos aspectos del código. *Master Search* está llena de formularios ya que hay que crear el árbol de decisión y cada uno de los candidatos que intervienen en la decisión. La etiqueta habitual en HTML para este tipo de componentes es <***form***></***form***>, angular posee una directiva, <***ng-form***></***ng-form***> que me facilita la implementación a la hora de las validaciones de los formularios con el método ***$valid*** sobre el ***ng-model*** de esta directiva.

Existen además etiquetas que no tienen nada que ver con el lenguaje HTML como son ***ng-controller*** que puede usarse como atributo en cualquier etiqueta del HTML, para definir que en esa etiqueta y resto de etiquetas anidadas la aplicación web va a trabajar sobre el controlador que se especifique en esta etiqueta. O ***ng-repeat*** para el lado de los controladores que me permite hacer un bucle de la variable que he pasado del controlador a la vista, por ejemplo los candidatos, en la vista hago un ***ng-repeat*** del array de candidatos para ir listando todos los que ha incluido el usuario.

Filtros

Los filtros permiten modificar el modo en el que se muestra la información en la interfaz de la aplicación al usuario. Angular cuenta con una cantidad de filtros ya creados para su uso directamente que pueden facilitar la programación.

Yo he utilizado un filtro únicamente para la depuración del código o cuando quería imprimir algo en la interfaz y comprobar si todo se esta guardando adecuadamente a la vez que introduzco los datos.

Por ejemplo, para saber si se crea correctamente el array de candidatos cuando introduzco un nuevo candidato, en la vista del ***sidenav-right*** introduzco un filtro y a la vez que relleno el formulario compruebo que los inputs se van introduciendo como deben.

{{candidatos | json}}

Scope

Los scopes son un núcleo fundamental de cualquier aplicación de Angular. Se utilizan a lo largo de todo el framework ya que son los que nos permiten establecer la unión entre las vistas y los controladores, permitiéndome pasar los datos de una manera bidireccional.

Justo antes de que la aplicación represente la vista en la interfaz del usuario, ésta se une al scope y la aplicación configura el DOM. Debido a la actualización del scope en tiempo real siempre se va a mantener actualizado como este la interfaz en cada momento, de esta manera te permite confiar en la información que contiene.

Cuando angular arranca y genera una vista, creará la unión entre la directiva ***ng-app="MasterSearch"*** que se encuentra en el elemento DOM que engloba la app, y el $rootScope. Esta variable por asemejarla al lenguaje que tenemos habitualmente en java orientado a objetos es lo más parecido a una variable global. Se suele modificar el valor desde los controladores donde cada vez que se crea o edita un nodo del árbol de decisión o un candidato se introduce en el objeto correspondiente que a su vez se actualiza en el scope y se muestra desde el scope en la vista.

Controladores

Los controladores son los encargados de inicializar, actualizar y modificar los datos que contiene scope implementando en cada uno las funciones correspondientes descritas en la sección de identificación de componentes.

Para unir la vista con el controlador, angular permite incluir una directiva suya "***ng-click***"; esta directiva une el evento "clic con el ratón", el cual es el encargado de llamar a la función correspondiente.

La aplicación está llena de esta aplicación porque para cualquier acción se necesita un botón en la vista que utiliza esta directiva y llama a la función del controlador. Por ejemplo al borrar el proyecto, es un botón con la siguiente etiqueta:

<button ng-click="deleteProyect(id)">

Contiene ***ng-click*** con la llamada al método ***deleteProyect(id)*** el cual llama al controlador navbar.js que contiene la implementación de dicha función. En primer lugar abre un modal de confirmación de la acción y seguidamente si el usuario confirma se procede a eliminar los datos de la aplicación para crear un nuevo proyecto desde cero.

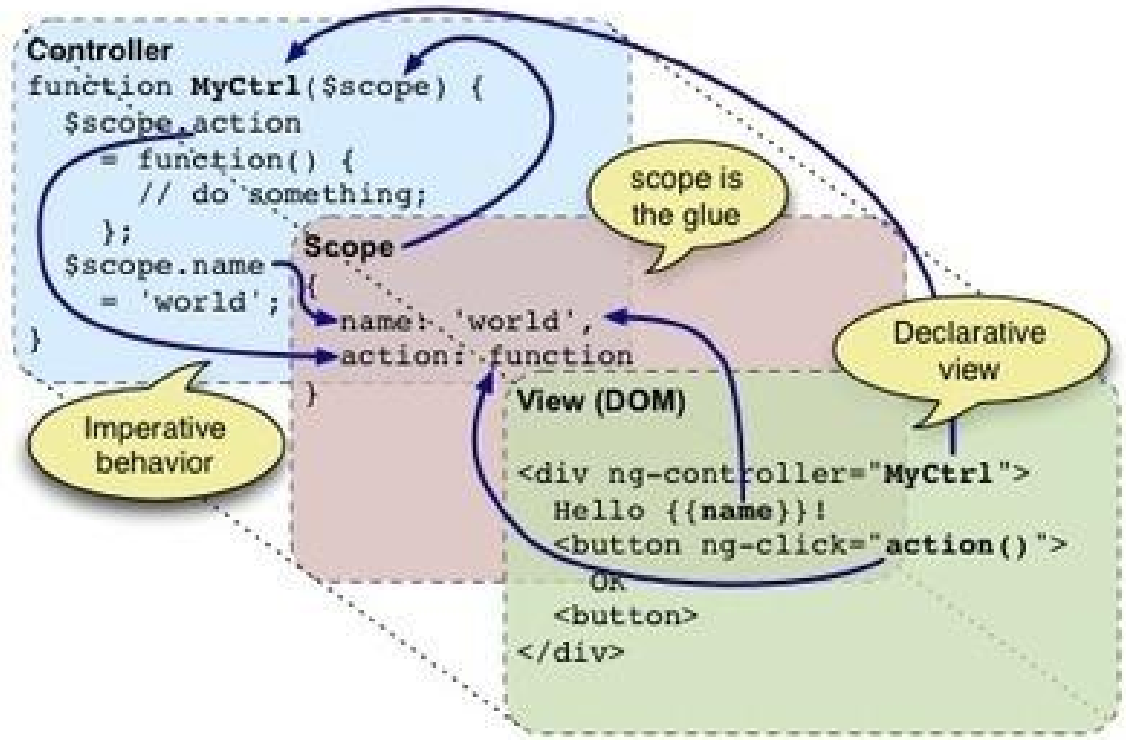


Ilustración 13: Funcionamiento de la arquitectura

Después de haber visto las vistas, controladores y scope adjunto esta imagen como resumen y ejemplo del funcionamiento de la arquitectura empleada para la herramienta de decisión multicriterio

Factoría

Las factorías son como contenedores de código que podemos usar en nuestros sitios desarrollados con AngularJS. Son un tipo de servicio con el que podemos implementar librerías de funciones o almacenar datos.

Angular consigue ese comportamiento usando el patrón "***Singleton***" que quiere decir que cada vez que se necesite ese objeto se enviará una instancia de ese objeto en lugar de instanciar de nuevo un ejemplar.

En este proyecto no ha sido necesario utilizarlo peo está bien tenerlo en cuenta en la arquitectura para futuras optimizaciones.

Servicios

Los servicios son los encargados de comunicarse con el servidor para enviar y obtener información que después será tratada por los controladores para mostrarla en las vistas.

Con el fin de mantener datos permanentemente durante todo el ciclo de vida de la aplicación, como por ejemplo los datos de cada uno de los nodos del árbol de decisión, AngularJS hace uso de los servicios.

Utilizo un ***servicioMain*** que me hace de puente entre todos los controladores. Por ejemplo, cuando creo un candidato desde el controlador ***sidenav-left*** llamo al controlador ***sidenav-right*** pasándole como parámetros el id del candidato que desea el usuario editar, dicho controlador se encarga de abrir el menú derecho con los datos

del candidato y cuando realice el submit el usuario se manda los datos al otro controlador, todo esto se hace de forma sencilla gracias al servicio.

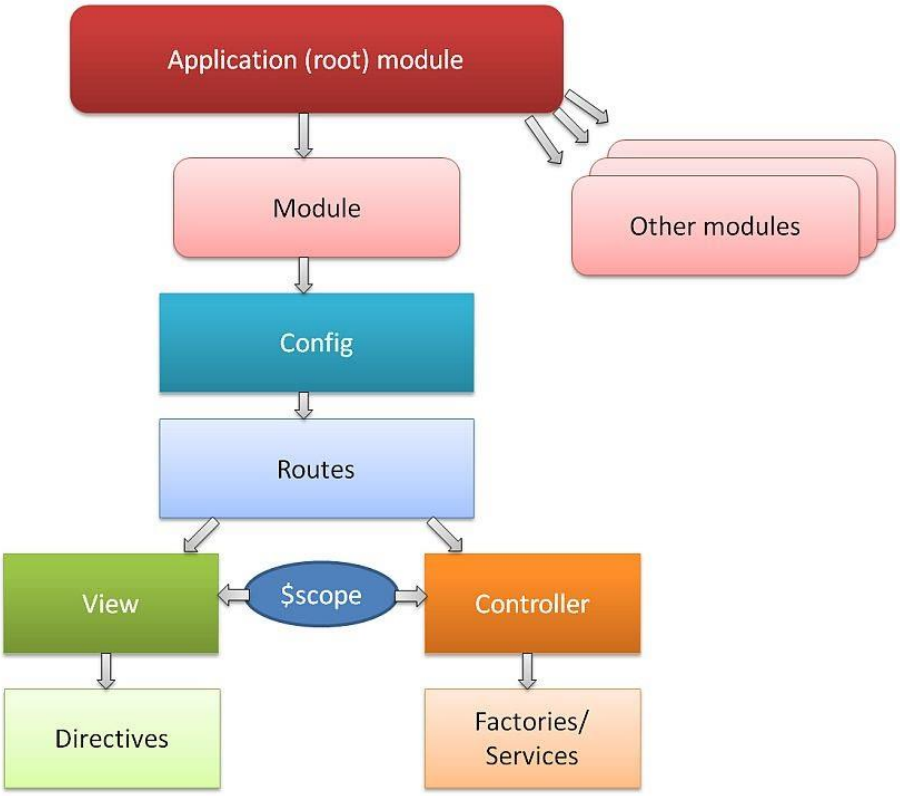


Ilustración 14: Arquitectura global

Para finalizar la arquitectura de AngularJS elegida adjunto a modo resumen el esquema de la ilustración anterior. Aquí quedan todos los elementos relacionados y ordenados.

MVC

En AngularJS el patrón que se implementa en la arquitectura es el del Modelo Vista controlador, desarrollando la vista en HTML y el modelo y controlador en javaScript.

Ya he entrado en detalle de cada uno de los componentes dentro del diagrama de la arquitectura que he implementado en el proyecto, a continuación adjunto una imagen a modo esquema del patrón MVC.

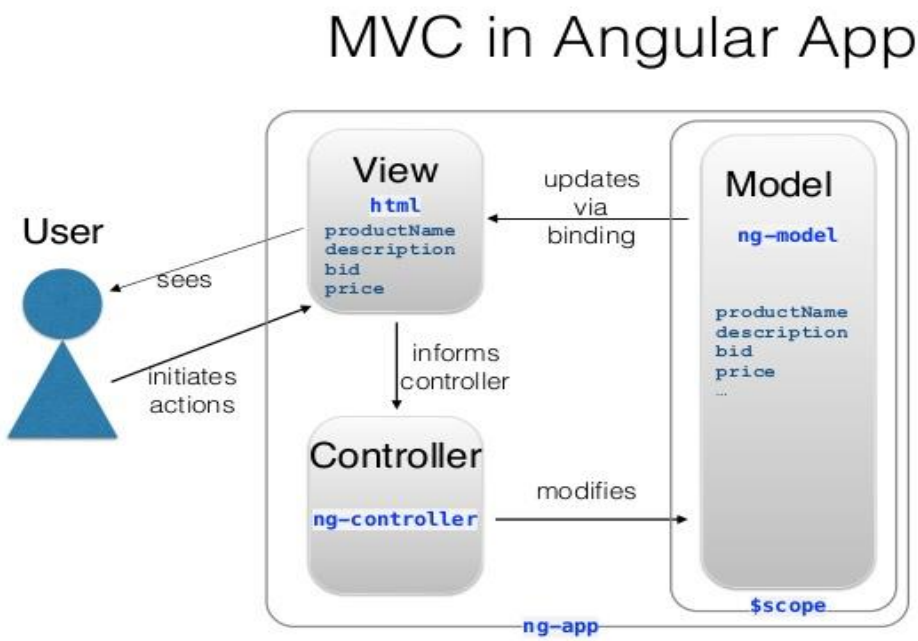


Ilustración 15: MVC

En función a este patrón es en el que me inspiro para crear la lógica de carpetas dentro del proyecto organizándolas por los diferentes componentes y en cada uno de ellos la vista, controlador y modelo.

Para ver la organización de los directorios adjunto un diagrama de mis directorios en el visualStudio.

He establecido un color a cada componente, para que se vea la arquitectura de modelo-vista-controlador para la separación y organización de carpetas pero dentro de las está la lógica de componentes de AngularJS.

Cada componente se divide en vista, modelo (style) y controlador, dashboard (naranja), navbar (azul), sidenav-left (verde), sidenav-right (morado). En la carpeta services solo tengo un servicio main que es el puente entre cada uno de los controladores y el index.html es el archivo HTML encargado de inyectar el resto de vistas del proyecto.

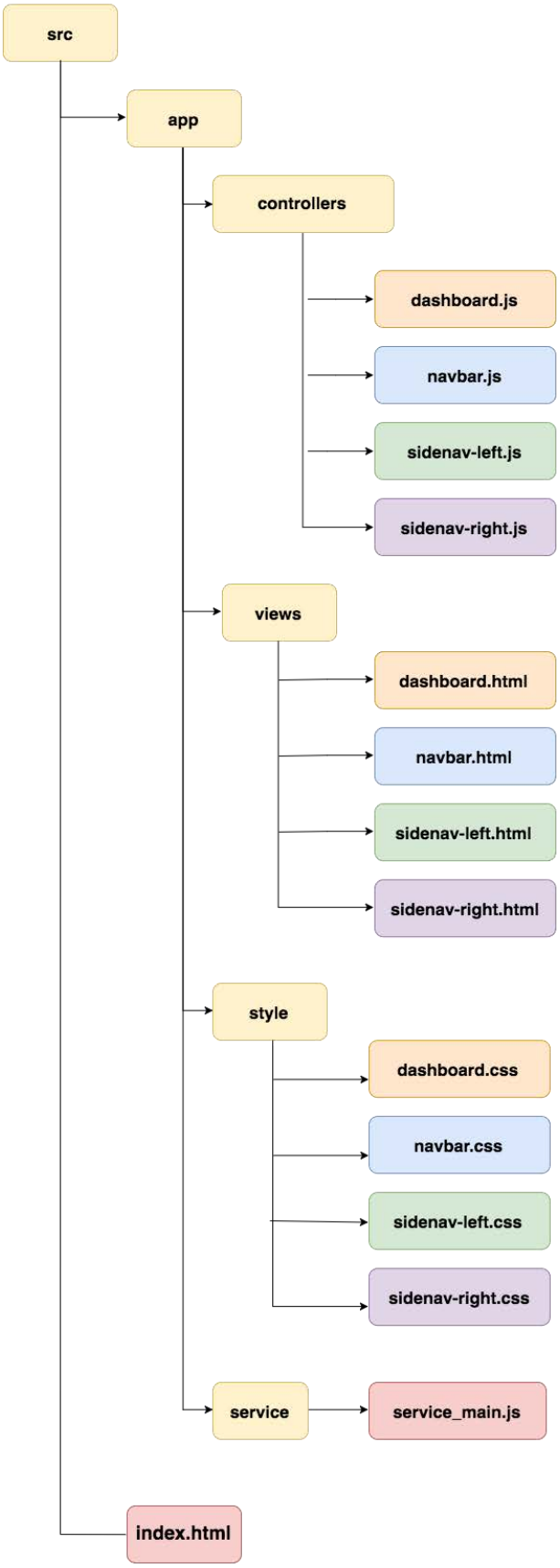


Ilustración 16: Directorios

5.2 Diseño de la interfaz

Como he comentado a lo largo del documento la interfaz que ve el usuario de esta aplicación es el pilar base para hacer un buen trabajo.

Uno de los objetivos principales por petición del cliente es lograr una interfaz usable que haga que el usuario la primera vez que se ponga delante de la pantalla le resulte sencilla de utilizar y transmita ganas de empezar a hacer el proyecto

Antes de empezar a prototipar la interfaz y a hacer la maquetación de la aplicación he fijado unos principios de diseño que deben cumplir todas las pantallas para lograr la mejor experiencia de usuario posible.

Voy a dividir esta sección en dos subsecciones en una explicaré los principios de diseño que sigo para el diseño con una breve descripción y en la otra adjuntaré prototipos de Sketch de la interfaz tal y como se ha pensado que ha sido desarrollada acompañados de explicaciones para reforzar el contenido.

5.2.1 Principios de diseño

Control del usuario

El usuario debe conocer en todo momento el lugar dónde se encuentra, con esto quiero decir que debe conocer en qué punto de la aplicación está. Con esto le permite conocer los pasos y acciones que puede hacer en ese momento y a las que puede llegar si acciona alguna de ellas.

En este principio también se debe tener en cuenta que el usuario debe sentir que tiene el control sobre la aplicación. Por ejemplo, cuando quiere realizar una acción general que se encuentra en el navbar se le mandará un modal de confirmación ya que son acciones que afectan a todo el proyecto y tienen una repercusión importante.

Forma parte del feedback de las acciones que hace, si le da a eliminar proyecto se le avisará que lo va a eliminar si quiere continuar y la repercusión que ocasiona (feedback).

Consistencia

La aplicación debe tener una consistencia importante, este es uno de los principios fundamentales de la experiencia de usuario ya que siempre debe tener el mismo layout la app para que cuando navegues por ella no da la sensación en pantallas sueltas que te has salido de la app, así como los colores deben ser unificados creando un tema común, la tipografía, es decir, la línea de estilo deber ser fija.

Intuitivo

"El usuario tiene que encontrar las cosas donde espera encontrarlas". Con esta frase se resume el principio de predicción de una interfaz. Los usuarios están preparados para localizar cosas en la ubicación habitual de estas, es decir, como norma estándar de las aplicaciones así como de las páginas web, se espera en la esquina superior izquierda el logo acompañado de un icono de retorno a la página "home" de la interfaz.

En los prototipos que he diseñado, y por tanto en la aplicación final sitúo las acciones donde se espera tenerlas, para editar le asigno el icono del lápiz de edición para eliminar la papelera, estos son algunos ejemplos del cumplimiento de este principio.

Sencillez

Este es uno de los principios más repetidos a lo largo del documento, desde un inicio busco la limpieza, simplicidad y sencillez de la interfaz. Se trata de un algoritmo algo complejo de implementar pero que debe ser sencillo manejar desde esta interfaz diseñada para ello.

Establezco un layout con un header azul y un background gris con textos sencillos y poco abundantes, todo muy intuitivo de cara al usuario y sin sobrecargar demasiado el dashboard para que de una mirada se quede todo visto.

Personalidad

En las app web que diseño e implemento me parece muy importante que tengan un toque de personalidad lo que le aporta seguridad a la interfaz, y una filosofía equilibrada a lo largo de todas las vistas.

Esta en concreto debe desprender sencillez de utilizar y un toque atractivo ya que los usuarios a los que va destinado no son técnicos (ingenieros), sino que en principio lo utilizará todo tipo de usuario que no posee una formación informática y debe estar a su alcance, como gmail que nadie se encuentra problemas a la hora de utilizar sus funciones básicas.

5.2.2 Interfaz

En esta subsección se recoge todos los prototipos realizados en sketch de la interfaz para ir analizando y explicando uno a uno sin mucho detalle las decisiones tomadas y los principios de la sección anterior que aplican.

Para comenzar realice unos prototipos básicos sobre papel con un bolígrafo de las secciones que contendría la aplicación. A continuación inserto una imagen de una fotografía del primer diseño sobre los apartados que tendría que tener.

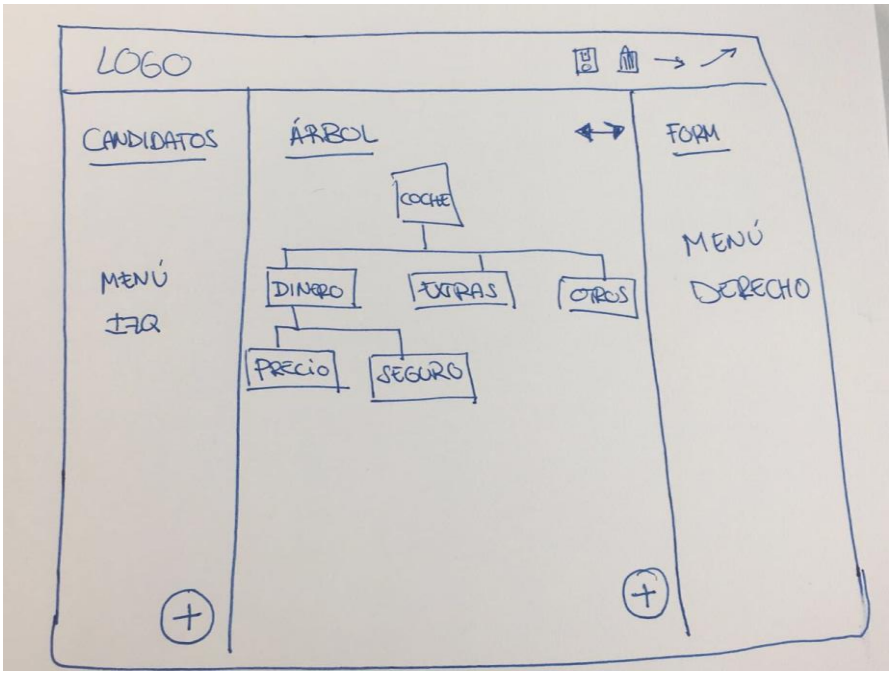


Ilustración 17: Maquetación inicial sobre papel

La idea es dividir la pantalla en dos grandes zonas: header (navbar) y la pantalla principal (dashboard, sidenav-left y sidenav-right).

- Zona 1: tenemos dos contenidos en esta zona, uno alineado totalmente a la izquierda con un padding y otro a la derecha que se trata de acciones sobre el proyecto completo. A la izquierda se sitúa el icono de home y el nombre de la app web; y a la derecha las acciones de guardar proyecto, eliminar proyecto y todas las necesarias de los requisitos del cliente.

- Zona 2: en esta zona tienen que entrar tres divisiones, el dashboard general, el menú izquierdo y el menú derecho. Dado que el dashboard contiene el árbol de decisión del usuario se considera la parte más importante y que debe ocupar la mayoría de la pantalla por esto, decidí que el menú lateral derecho fuese dinámico y aparezca solo en caso de las acciones de los candidatos. El menú izquierdo sí que es fijo porque ahí aparece otra parte importante de la herramienta que son los candidatos a la elección. Como se puede observar el dashboard contiene el árbol con crecimiento horizontal, es la primera idea pero en los prototipos de Sketch se ve que esa decisión la tuve que variar ya que si se trata de un árbol grande con muchos nodos y niveles el usuario no lo ve de un vistazo y diseñe un árbol con crecimiento vertical.

Una vez analizado el prototipo que realice sobre papel para fijar una idea de lo que había que incluir en la interfaz, para ver detalles y entrar en cada subzona más en detalle, empiezo a ir adjuntando los diseños realizados en Sketch. Esta aplicación me permite acercarme mucho a la aplicación programada, es prácticamente capturas de pantallas de la app ajustado al pixel y si se incluyen interacciones de InVision el cliente puede valorar si se entiende y lo aprueba o no cumple las necesidades de los usuarios finales.

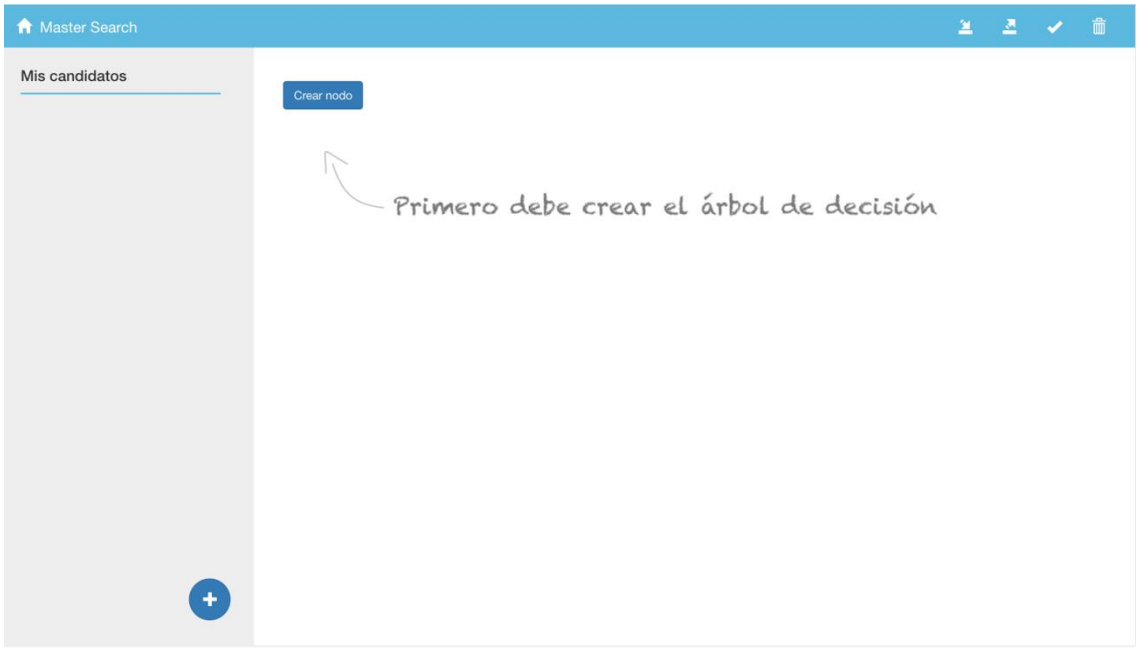


Ilustración 18: Prototipo pantalla general

Prototipo de la pantalla principal cuando se abre la aplicación en la que se diferencia las tres zonas a priori estáticas y en la que se especifica al usuario que lo primero que debe hacer es crear el árbol de decisión para poder realizar otro tipo de acciones.

Se muestra una clara aplicación de los principios de diseño comentados anteriormente ya que se sigue una línea limpia y simple, en la que de primeras no tiene mucha información para no causar confusión al usuario sino que es claro lo que debe hacer en primer lugar.

Resalto el uso del color azul oscuro con bordes redondeados para las acciones que repercuten sobre un componente como es “crear nodo”, y la tipografía de color blanca para que resalte sobre el fondo oscuro.

Otra apreciación es el color de la cabecera: un azul claro que resalte y de personalidad a la aplicación pero pase desapercibido frente a las acciones y el objetivo de la herramienta; así como el menú izquierdo con un fondo gris claro para diferenciarlo del dashboard pero que no llame demasiado la atención.

Dashboard

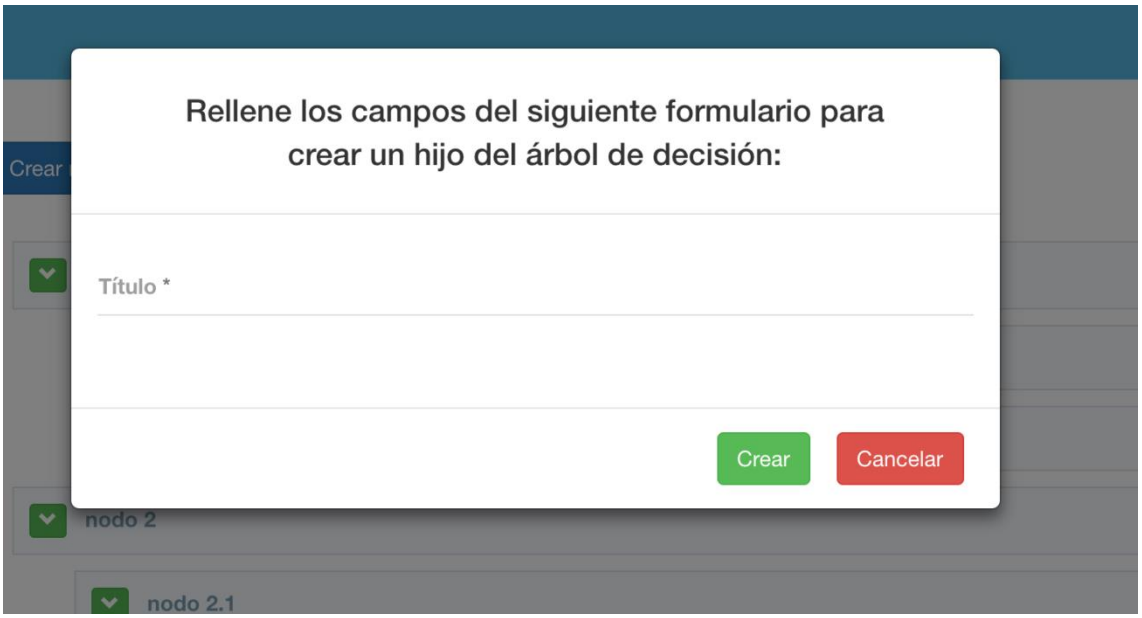


Ilustración 19: Añadir un criterio

El primer paso como indica la primera pantalla es crear nodos para ir creando el árbol de decisión por lo que al crear un nodo pensé en lanzar un modal en el que se fija el nombre del criterio, dado que no tiene muchos inputs el formulario puede lanzarse en un modal de este tipo y el usuario no pierde la información de fondo dado una sensación de seguridad y de reflejo automático nada más guardar del nuevo criterio en el árbol.

Una vez creado el nodo se pinta en el árbol con un formato de caja para especificar que las acciones que aparecen a la derecha son sobre ese nodo: editar, eliminar o añadir un nodo hijo.

Ilustración 20: Árbol de decisión

Se observa que continúo con el principio de consistencia mostrando los botones que realizan acciones igual que el estilo que he descrito para los que aparecen en la pantalla principal.

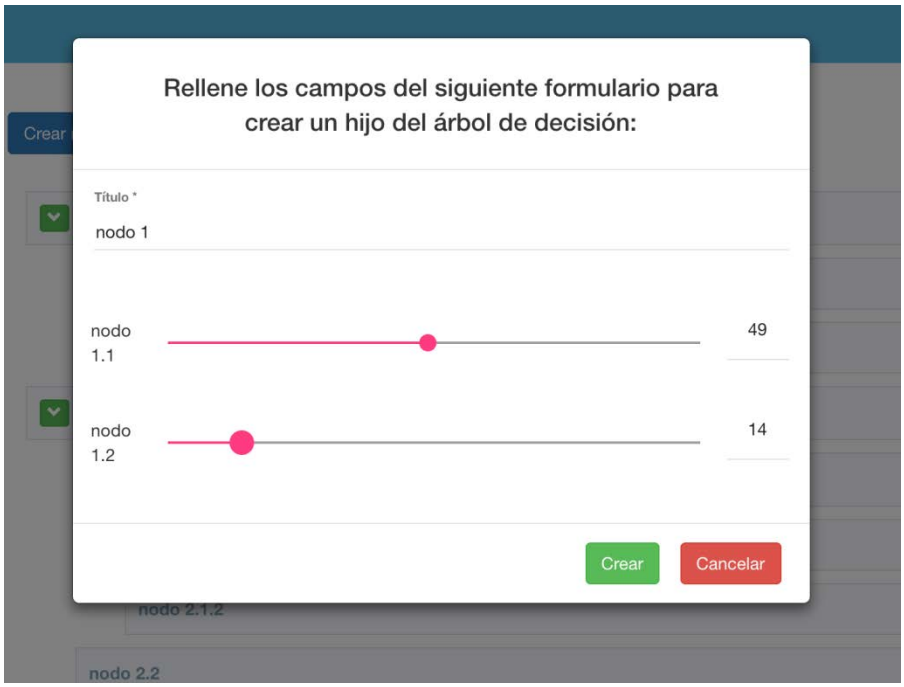


Ilustración 21: Añadir criterio

En el diseño un requisito difícil de reflejar de una manera sencilla e intuitiva fue establecer los pesos por pares de los nodos hijos de cada uno de los criterios padres, la mejor opción que encontré fue que al editar te salgan esos campos para fijarlos si se cumplen las precondiciones dado que tenía que ser un popup que no ocupase espacio fijo en la pantalla pero que se pueda consultar en cualquier momento para modificarlo.

Este diseño es el final del árbol de decisión que luego he implementado porque como he comentado en el prototipo en papel, el árbol en un principio lo pensé con un crecimiento horizontal pero en el prototipo me di cuenta que en el caso extremo que tiende a infinitos nodos habría que meter un scroll tanto horizontal como vertical lo que hace que la experiencia de usuario disminuya notablemente.

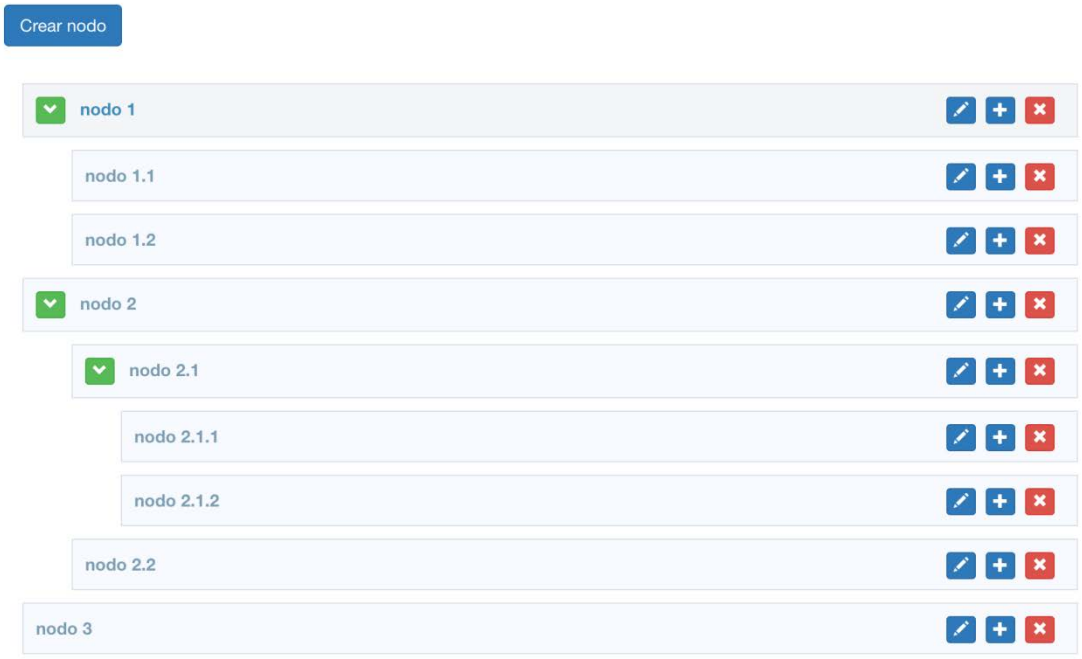


Ilustración 22: Árbol de decisión

Es aquí donde la implementación de un árbol con crecimiento vertical, me permite realizar acciones sobre cada uno de ellos o incluso colapsar/expandir los nodos anidados de un nodo padre para visualizarlo mejor.

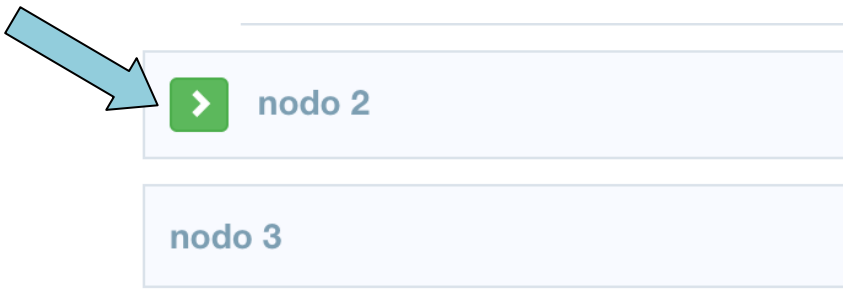


Ilustración 23: Expandir nodo

Sidenav-left

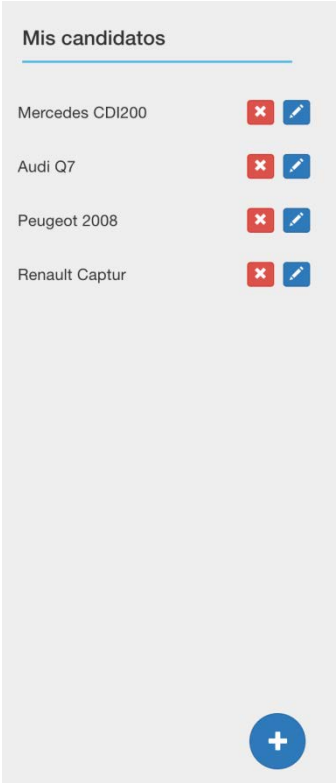


Ilustración 24: Prototipo lateral izquierdo

Una vez creado el árbol de decisión con los criterios que el usuario considere relevantes en la decisión, se debe pasar a añadir los candidatos. Denomino candidatos a las opciones posibles que barajea el usuario para su decisión.

Estos candidatos serán listados, una vez añadidos, en el menú fijo lateral izquierdo que se les identificará en la interfaz por el nombre que establezca el usuario. Para añadir o editar un candidato se despliega el menú lateral derecho que veremos en la sección de sidenav-right. En esta sección se muestra un prototipo de la lista de candidatos y de las acciones que se pueden realizar sobre cada uno de ellos.

Se aprecia un título del menú subrayada de una manera sutil por una línea horizontal de color corporativo y una lista con el nombre y a la derecha las acciones que puede realizar. En el último candidato vemos el efecto debe realizar la aplicación cuando el ratón pase por encima del candidato (“hover”).

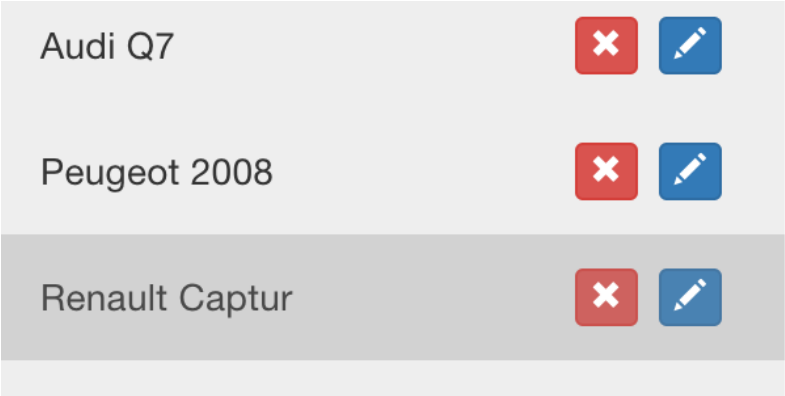


Ilustración 25: Candidatos

Esta imagen es una captura dentro del prototipo de la lista de candidatos del menú izquierdo para que se pueda observar de cerca la consistencia que se mantiene entre los nodos con sus respectivas acciones y los candidatos.

La diferencia con los criterios es que estos últimos no se pueden ordenar ya que es una lista indiferentemente del orden y no se pueden añadir candidatos anidados, es decir, tienen las funciones de editar y eliminar comunes y no tiene las propias de un árbol.

Sidenav-right

Única zona de la aplicación que no es fija, para no invadir espacio visual del árbol de decisión mostrado en el artboard, decidí tras prototipar varias opciones que este menú se mostrará sólo cuando se pulse el botón de añadir candidato o el de editar.

Esto se debe confirmar con el cliente y una vez aceptada ya se pasará al desarrollo.

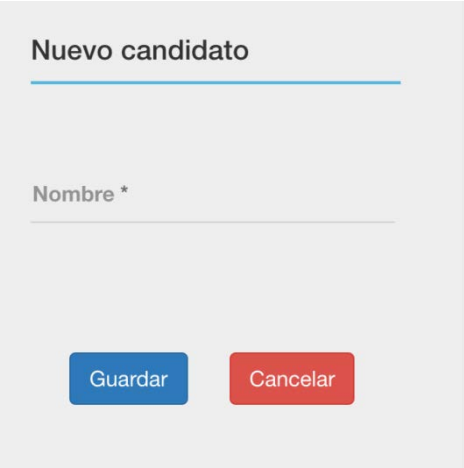


Ilustración 26: Prototipo menú lateral derecho

Se trata de un formulario que se completa con los criterios hijos del árbol, se deben rellenar todos los campos por cada uno de los candidatos y hacer submit. Los input siguen el diseño de limpieza simple como los el modal y los colores del título y línea horizontal inferior se mantienen. En el prototipo hice un sólo input de ejemplo, en la aplicación final se mostrarán tantos como sean necesarios.

Navbar

Header siguiendo el estándar de aplicaciones, a la derecha accione generales y a la izquierda título de la app. Este header se mantiene a lo largo de toda la app, destacar que los botones para acciones generales los he diseñado con el icono sin fondo por cada botón dado que ya tenemos un fondo en el header y uno de los principios era la sencillez, no sobrecargar innecesariamente vistas.

Ilustración 27: Prototipo navbar

Resultados

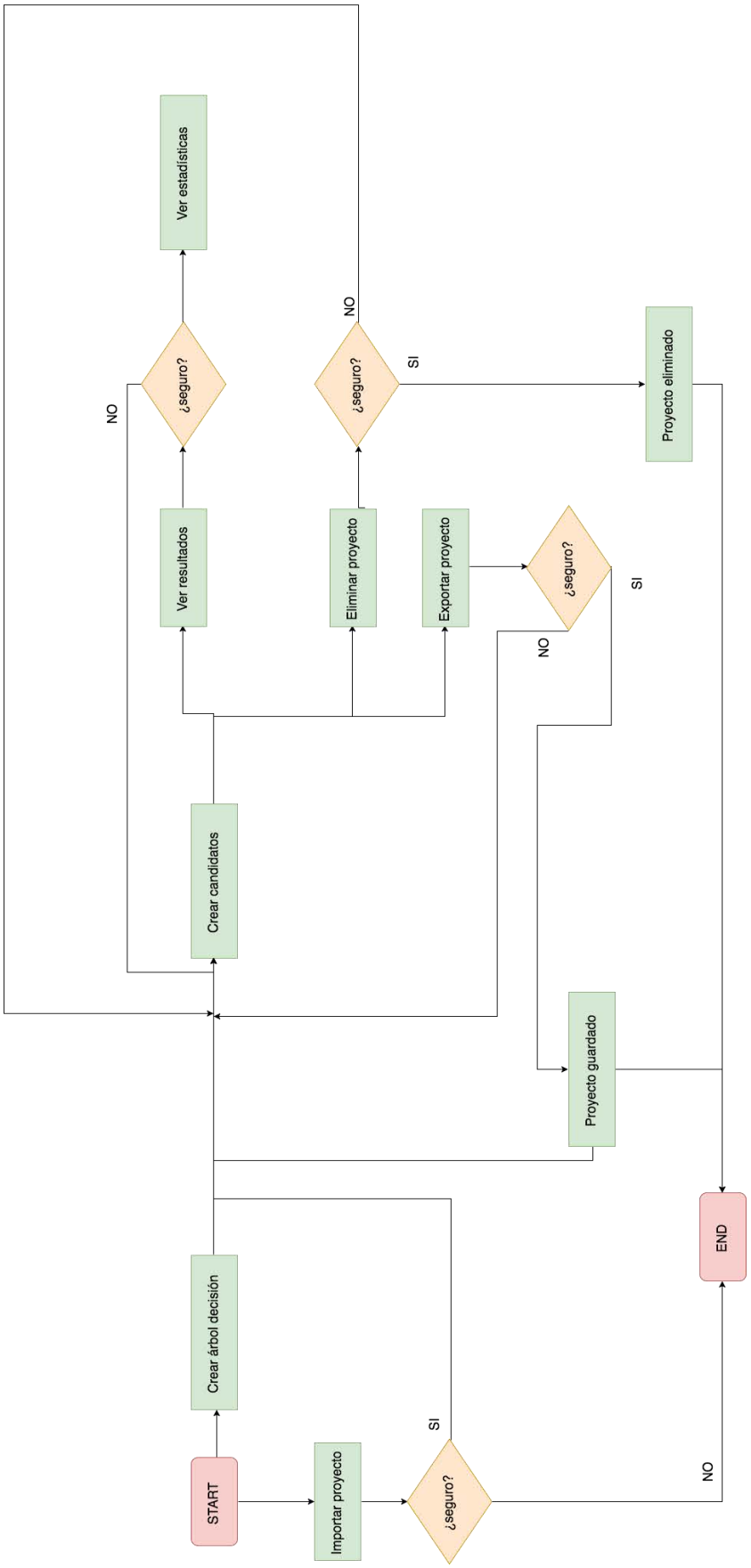
Esta pantalla mantiene el layout de la aplicación, es decir el header y el fondo, pero en el dashboard se muestran gráficos correspondientes a las estadísticas sacadas con el algoritmo AHP multicriterio. Se resalta el candidato elegido y el orden en los que sería conveniente elegir si hay que coger los tres primeros o x en determinadas situaciones.

5.3 Conclusiones

En esta sección se expone tanto el lado de arquitectura del sistema lo que pertenecería a un diseño de la aplicación por dentro y otra sección de diseño de la interfaz dónde se adjuntan prototipos y capturas del diseño realizado por Sketch previo al desarrollo.

Este capítulo es extenso en el diseño de la interfaz porque el proyecto tiene una gran carga visual demandada por parte del cliente y necesaria para lograr los objetivos por lo que he querido explicar todo al detalle. Por esta razón este apartado se podría considerar una guía de ayuda en caso de que el usuario no sepa cómo realizar una acción nada más abrir la app.

Para terminar el capítulo inserto un diagrama de flujo de un proyecto en Master Search para que se fijen las ideas del diseño y más que nada, como he dicho este capítulo sirve como manual de usuario por lo que en el siguiente diagrama puede encontrar los pasos que debe seguir a nivel general.


START
Importar proyecto
¿seguro?
SI
NO
Crear árbol decisión
Crear candidatos
Ver resultados
Ver estadísticas
¿seguro?
NO
Eliminar proyecto
¿seguro?
SI
Exportar proyecto
¿seguro?
SI
NO
Proyecto eliminado
Proyecto guardado
END

6. Pruebas

6.1 Introducción

Con el objetivo de asegurar que el software creado cumple con las especificaciones requeridas obtenidas en la fase de análisis, he diseñado y realizado una serie de pruebas que a su vez sirven para detectar y eliminar errores que no se hayan detectado previamente.

En este capítulo se recogen las principales pruebas funcionales que se han realizado sobre los casos de uso establecidos en el capítulo de análisis del sistema. Todas ellas se han evaluado de forma manual, es decir, las ejecuta un tester como si fuese un usuario normal, siguiendo el plan de pruebas diseñado en la fase de análisis de los requisitos con el objetivo de garantizar que la aplicación hace lo que debe.

En el caso de que alguna prueba no cumpla con la salida prevista se marca como rechazada y después de la corrección del código se vuelve a realizar hasta que todas tengan el estado completada. Para que no quede demasiado engorroso, en este documento simplemente se exponen una prueba por cada caso de uso con una plantilla de cómo se realizarían cada una de ellas y de esta forma hacer todas.

Las pruebas se realizan en un archivo test en cada componente que recoge una parte de define dónde inicializa las precondiciones necesarias para la prueba y después

va lanzando prueba a prueba. Este lanzamiento se realiza con karma, se encarga de ejecutar los test de Javascript según se vayan construyendo, de tal forma que ante cualquier fallo el desarrollador se dará cuenta de inmediato. Para la instalación de karma es necesario node.js que como ya quedó comentado en el capítulo de tecnologías es de lo primero que se instala.

6.2 Plan de pruebas

Por cada caso de uso y requisito de software identificado se realizará una prueba para comprobar que se ha alcanzado el objetivo requerido del cliente o del analista. Cada una de las pruebas queda descrita en una tabla con la información necesaria. A continuación se muestra una plantilla de las pruebas y se describe cada uno de los campos que contiene.

Caso de uso					
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**

Tabla 68: Plantilla de pruebas unitarias

- **Caso de uso:** se especifica el caso de uso que se va a probar junto su identificador único para localizarlo.
- **Id:** identificador inequívoco del caso de prueba. La sintaxis será siempre siguiendo este patrón **P_XX**.
    - **P:** sigla de prueba que identifica esta como prueba y la diferencia de caso de uso (CU) y de requisitos funcionales (RF) o no funcionales (NF).
    - **XX:** número de dos cifras con un orden cronológico de creación.
- **Descripción:** se define lo que se pretende verificar al lanzar la prueba, es decir el objetivo que se alcanza si pasa la prueba satisfactoriamente.
- **Prerequisitos:** estado de la aplicación antes de lanzar la prueba, condiciones que se deben haber cumplido para poder ejecutar la prueba.

- **Resultado esperado:** previsión de la salida en caso de que salga como se espera.
- **Resultado obtenido:** salida real al terminar la ejecución.
- **Estado:** indica el estado en cada momento de la prueba, puede tomar dos valores: completada o rechazada.

*En este documento solo se muestra una fila de ejemplo por cada caso de uso pero tendrán n filas, siendo n el número de pruebas necesarias para probar esa funcionalidad.

Caso de uso			CU_01: Crear nodo		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_01	Se debe comprobar que cuando no se introduce un texto en el input del formulario no te permite guardar.	Abrir la app web y darle a crear nodo.	El input se pondrá en rojo y no guardará el nodo ni desaparecerá el modal	No permite guardar el nodo indicando al usuario el input en rojo.	Completada

Tabla 69: PU_01

Caso de uso			CU_02: Editar nodo		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_02	Se debe comprobar que si se modifica el título del criterio en la pantalla se actualiza con el nuevo valor.	Abrir la app web, crear un nodo y pulsar el botón de editar dentro del nodo.	Se modifica el nodo dentro del árbol de decisión, comprobando que el título ha cambiado igual que en el array de node.	Guarda el título nuevo del nodo, actualizando el array[]node y comprobando la vista se ha cambiado.	Completada

Tabla 70: PU_02

Caso de uso			CU_03: Eliminar nodo		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_03	Se debe comprobar que si se elimina un nodo del árbol de borra de la vista con sus atributos y del array.	Abrir la app web, crear un nodo y pulsar el botón de borrar dentro del nodo.	Se elimina el nodo tanto del array en la posición indicada como de la vista	Borra el nodo, actualizando el array[]node sin la nueva posición y comprobando la vista se ha cambiado.	Completada

Tabla 71: PU_03

Caso de uso			CU_04: Ordenar nodo		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_04	Se debe comprobar que un nodo se coloca dentro de otro al moverlo a dicha posición.	Abrir la app web y crear dos nodos.	Se ordena un nodo arrastrándolo dentro de otro y comprobando que tanto en el array se introduce como dodo hijo del anterior como en la vista.	Ordena el nodo tanto en el array dentro del padre como en el árbol que se muestra en la vista.	Completada

Tabla 72: PU_04

Caso de uso			CU_05: Añadir nodo hijo		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_05	Se debe comprobar que al pulsar sobre añadir dentro de un nodo, el nuevo nodo se anida dentro del anterior.	Abrir la app web, crear un nodo y pulsar el botón de añadir dentro del nodo.	Se añade un nodo hijo del nodo dónde se pulsa el botón añadir. Se introduce el nuevo nodo	Nuevo nodo añadido en el array[]nodes del padre y se muestra en la vista anidado del padre para una	Completada

			dentro del array de nodos hijos del padre.	mejor visualización.	

Tabla 73: PU_05

Caso de uso			CU_06: Establecer un peso a sus nodos hijos.		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_06	Se debe comprobar que cuando un nodo tiene dos hijos al editar nodo se le permite al usuario establecer pesos por pares.	Abrir la app web, crear un nodo y crear dentro de ese nodo dos nodos hijos.	Cuando el usuario pulsa editar sobre el nodo padre se muestra un slider por pares	El usuario edita un nodo padre de dos hijos y se muestra en el modal un slider comparativo de pesos.	Completada

Tabla 74:PU_06

Caso de uso			CU_07: Introducir los candidatos para la decisión.		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_07	Se debe comprobar que cuando se pulsa el botón de añadir del menú lateral izquierdo se abre el menú lateral derecho para añadir candidato.	Abrir la app web, crear el árbol de decisión, establecer pesos y pulsar el botón de añadir candidato.	Cuando el usuario pulsa añadir se abre el menú lateral derecho.	El usuario pulsa añadir saliendo por el lateral derecho el menú de añadir candidato.	Completada

Tabla 75: PU_07

Caso de uso			CU_08: Editar candidato		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_08	Se debe comprobar que una vez realizado el submit los datos alterados han sido modificados en el candidato.	Abrir la app web, crear el árbol de decisión, establecer pesos, crear un candidato y pulsar sobre el botón de editar	Cuando el usuario pulsa guardar del formulario de edición los atributos se ven alterados.	El usuario pulsa guardar en el formulario y se actualizan los datos del candidato seleccionado.	Completada

Tabla 76: P_08

Caso de uso			CU_09: Eliminar candidato		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_09	Se debe comprobar que se borra del listado de la vista de candidatos.	Abrir la app web, crear el árbol de decisión, establecer pesos, crear un candidato y pulsar sobre el botón de eliminar.	Cuando el usuario pulsa eliminar desaparece el candidato de la lista del menú lateral.	El usuario pulsa eliminar candidato y se suprime del listado de todos los candidatos.	Completada

Tabla 77: P_09

Caso de uso			CU_10: Importar proyecto		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_10	Se debe comprobar que al pulsar sobre el botón de importar aparece un modal para adjuntar un	Abrir la app web, pulsar importar proyecto.	Cuando el usuario pulsa importar, aparece un	El usuario pulsa importar y le aparece un campo para	Completada

	archivo local.		modal con un lector de directorios.	introducir el directorio del archivo.	

Tabla 78: P_10

Caso de uso			CU_11: Exportar proyecto		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_11	Se debe comprobar que al pulsar sobre la confirmación de exportar se descarga un archivo.	Abrir la app web, crear el árbol de decisión, establecer pesos, crear candidatos y pulsar sobre el botón de exportar.	Cuando el usuario pulsa exportar le sale un modal y al confirmar acción se descarga un archivo.	El usuario pulsa exportar y al confirmar se descarga automáticamente un archivo.	Completada

Tabla 79: PU_11

Caso de uso			CU_12: Eliminar proyecto		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**
P_12	Se debe comprobar que al pulsar sobre eliminar confirmándolo se borra el árbol de decisión completo.	Abrir la app web, crear el árbol de decisión y pulsar sobre el botón de eliminar.	Cuando el usuario pulsa eliminar le sale un modal y al confirmar acción se borra el árbol.	El usuario pulsa eliminar y al confirmar se suprimen todos los nodos del árbol	Completada

Tabla 80: P_12

Caso de uso			CU_13: Obtener resultados		
ID	**Descripción**	**Prerequisitos**	**Resultado esperado**	**Resultado obtenido**	**Estado**

P_13	Se debe comprobar que al pulsar sobre obtener resultados cambia la página que se muestra	Abrir la app web, crear el árbol de decisión, añadir los candidatos y una vez terminado el estudio pulsar el botón de obtener resultados.	Cuando el usuario pulsa obtener resultados, se cambia la interfaz a una pantalla con estadísticas.	El usuario pulsa obtener resultados y obtiene un dashboad con gráficas de los resultados obtenidos.	Completada

Tabla 81: PU_13

7. Planificación y presupuesto

7.1 Planificación

El proyecto ha tenido una duración global de 6 meses y medio, lo que es lo mismo que 141 días laborables. Teniendo en cuenta que la jornada laboral del calendario es de 6/7 horas diarias salen alrededor de 900 horas.

Se comienza por exponer cada una de las fases en las que está dividido el proyecto que son un total de seis, para posteriormente adjuntar el diagrama de Gantt donde se ve reflejado todo el ciclo de este proyecto.

Análisis e investigación de la competencia

Esta es una de las fases más largas de todo el proyecto, se trata de 33 días investigando acerca del mercado buscando aplicaciones similares a la propuesta solicitada por parte del cliente. Una vez se ha investigado lo suficiente se debe buscar la mejor herramienta y las tecnologías para desarrollar el proyecto de la mejor manera posible. Se incluye todo el proceso de aprendizaje de nuevas tecnologías y la instalación y preparación del entorno de desarrollo para empezar a implementar.

Análisis

Fase en la que se habla con el cliente y se analiza la propuesta de desarrollo con las posibles soluciones. Se sacan las características y restricciones que la aplicación debe tener, es decir, el documento de casos de uso y requisitos de software funcionales y no funcionales.

Diseño y arquitectura

En la tercera fase se pone en práctica los requisitos y conclusiones sacadas de la fase anterior de análisis con los documentos de casos de uso que empleo para diseñar la interfaz de usuario reforzando la experiencia. Antes del diseño de la interfaz web se realiza un diseño de la arquitectura a nivel de aplicación con las tecnologías estudiadas en la primera fase y en la configuración del entorno de desarrollo.

Desarrollo e implementación

Fase más larga del proyecto, tiene una duración global de 50 días con 6h diarias de trabajo en la que se implementa con el entorno ya montado y los diseños, tanto de arquitectura de sistema como de interfaz, toda la aplicación web. A modo resumen, se juntan todas las fases anteriores y se implementan para que funcione de manera automatizada y correctamente.

Pruebas

Tiene dos partes: en la primera, se planifican y redactan cada una de las pruebas unitarias de cada requisito y caso de uso para posteriormente en la fase dos se ejecuten y se implementen comprobando una a una su cumplimiento tanto por el lado que debe funcionar como comprobando el lado que debe fallar si falla.

Documentación

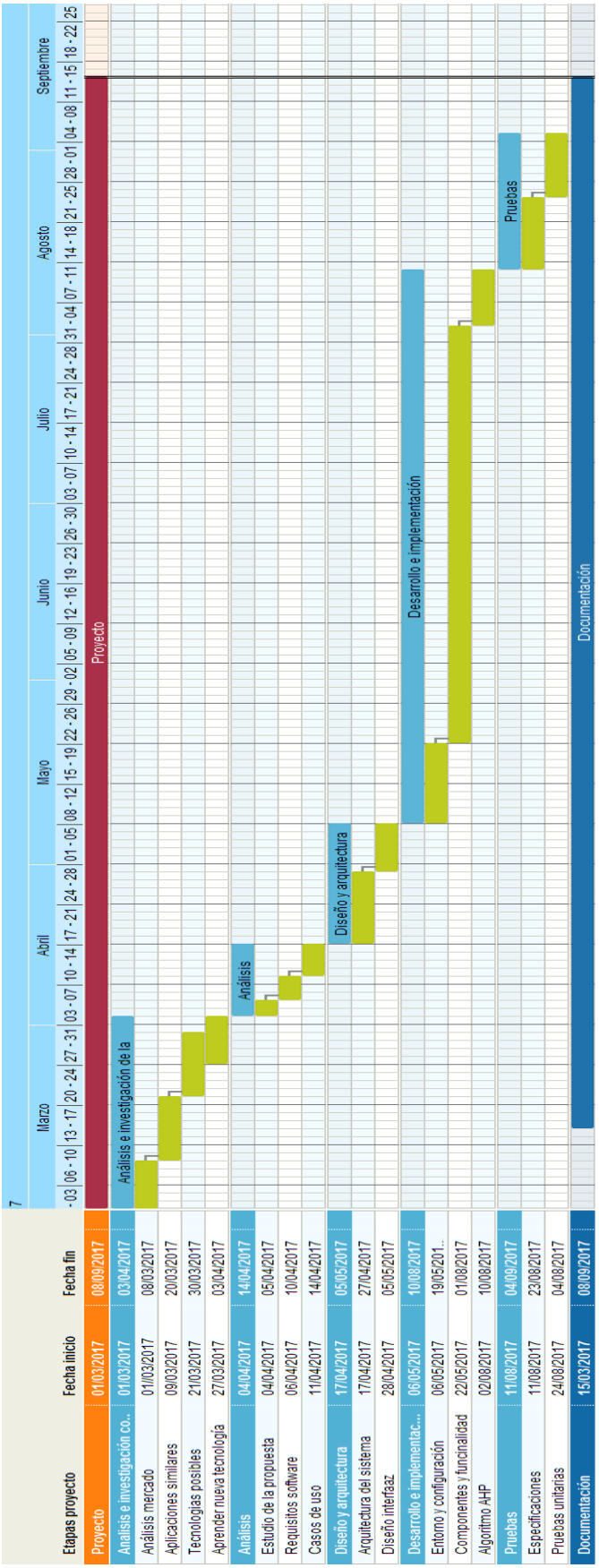
La fase de documentación es una fase que dura todo el proyecto pero que se realiza poco a poco. Se decidió redactar a la par que se desarrolla una media de 2h diarias y luego dedicar unos días finales a la revisión y corrección de fallos detectados o ausencia de apartados. En esta fase el objetivo es este documento, la memoria del proyecto final lo más concreta posible y útil para el cliente, técnicos u analista.

En la siguiente tabla se muestra un resumen de horas empleadas en cada fase con la correspondencia en días para tener una idea general de cómo se han repartido por las fases y cuál tiene más peso que otra.

Fases del proyecto	Horas
Análisis e investigación de la competencia	210
Análisis	55
Diseño	119
Desarrollo e implementación	290
Pruebas	85
Documentación	141
TOTAL	900

Tabla 82: Fases del proyecto

A continuación adjunto el diagrama de Gantt que me permite mostrar de forma gráfica las fases del proyecto.


Marzo
Abril
Mayo
Junio
Julio
Agosto
Septiembre
Etapas proyecto
Fecha inicio
Fecha fin
Proyecto 01/03/2017 08/09/2017
Analisis e investigación co.. 01/03/2017 03/04/2017
Análisis mercado 01//03/2017 08/03/2017
Aplicaciones similares 09/03/2017 20/03/2017
Tecnologias posibles 21/03/2017 30/03/2017
Aprender nueva tecnología 27/03/2017 03/04/2017
Análisis 04/04/2017 14/04/2017
Estudio de la propuesta 04/04/2017 05/04/2017
Requisitos software 06/04/2017 10/04/2017
Casos de uso 11/04/2017 14/04/2017
Diseño y arquitectura 17/04/2017 05/05/2017
Arquitectura del sistema 17/04/2017 27/04/2017
Diseño interfaaz 28/04/2017 05/05/2017
Desarrollo e implementac... 06/05/2017 10/08/2017
Entorno y configuración 06/05/2017 19/05/201...
Componentes y funcinalidad 22/05/2017 01/08/2017
Algoritmo AHP 02/08/2017 10/08/2017
Pruebas 11/08/2017 04/09/2017
Especificaciones 11/08/2017 23/08/2017
Pruebas unitarias 24/08/2017 04/08/2017
Documentación 15/03/2017 08/09/2017
Proyecto
Análisis e investigación de la
Análisis
Diseño y arquitectura
Desarrollo e implementación
Pruebas
Documentación

7.2 Presupuesto

El coste global del desarrollo del proyecto de una herramienta web para la resolución de problemas de decisión multicriterio con AHP lo calculo obteniendo los costes parciales relacionados con cada subcoste: de material, hardware, software y material de oficina, añadiéndole un porcentaje del 20% de costes improvisados e indirectos.

7.2.1 Coste de personal

Para la implementación de la aplicación y la documentación en este documento se ha decidido incluir dos roles diferentes.

El analista que se encargará de la fase de análisis e investigación previa del mercado donde se valora las app similares y las nuevas tecnologías que se pueden utilizar y posteriormente del análisis de los requisitos pedidos por el cliente para analizarlos y poder redactar los documentos de requisitos de software y casos de uso necesarios. El programador se encarga de la fase de diseño y arquitectura, el desarrollo del código junto a las pruebas funcionales y la documentación. Los dos roles cuentan con 2 años de experiencia como junior.

En la siguiente tabla se observa la carga de trabajo y el coste que ha supuesto en el proyecto cada recurso de personal por hora, obteniendo la suma total para poder luego hacer el coste global del proyecto.

Personal	Horas	Precio/hora	Total
Analista Junior	105	16,97	1781,85
Programador Junior	578	13,26	7664,28
			9446,13

Tabla 83: Presupuesto coste personal

Los precios que cada uno de los roles cobra por hora los he obtenido en la red sobre una media que he hecho sobre varias páginas encontradas en internet en la comunidad autónoma de Madrid. El coste del personal suma un total de 9446,13 sin IVA.

7.2.2 Coste hardware

En este apartado se calcula el coste de los equipos utilizados así como de todos los materiales hardware necesario para desarrollar el proyecto. Para calcular la amortización que se tiene de proyecto se utiliza la fórmula proporcionada por la universidad como guía.

(A/B)xCxD

A: número de meses desde la fecha de facturación en que el equipo es utilizado.

B: Periodo de depreciación = 60 meses

C: Coste del equipo

D: Porcentaje del uso que se le dedica al proyecto.

En la siguiente tabla se recogen cada uno de los equipos y material hardware utilizados junto con las unidades la dedicación y el coste final tras la aplicación de la formula anterior para sacar el coste real.

Hardware	Unidades	Coste	Dedicación	Coste final
Macbook Air 17”	1	1349	6 meses	134,9
Windows Surface Pro	1	1100	6 meses	110
Disco duro 1TB	1	109,99	6,5 meses	11,91
				256,81

Tabla 84: Presupuesto coste hardware

Tras realizar el cálculo el coste de hardware suma un total de 256,81 euros sin IVA incluido.

7.2.3 Coste de software

Ahora es el momento de sumar los gastos software del proyecto tanto de licencias como herramientas o cualquier elemento que haya sido necesario para el desarrollo. Casi todos los coste son de 0 euros respecto a librerías que se tratan de librerías Open Source.

Software	Unidades	Coste/unidad	Coste total
Visual Studio Code	1	0	0
Nodejs	1	0	0
Bower	1	0	0
Angular material	1	0	0
Bootstrap	1	0	0
Karma	1	0	0
Microsoft Office	1	177,75	177,75
Sketch	1	39,99	39,99
			217,74

Tabla 85: Presupuesto coste software

La suma total del software necesario suma 217,74 euros.

7.2.4 Coste de material de oficina

El último desglose de los costes viene del material de oficina, aquellos que al usarlo se van desgastando con el tiempo, material fungible.

Material de oficina	Unidades	Coste/unidad	Coste total
Cuaderno	1	6,35	6,35
Lápiz	1	0,3	0,3
Bolígrafo	1	0.50	0,50
Goma	1	0,20	0,20
Carpeta	1	5,25	5,25
			12,60

Tabla 86: Presupuesto coste material

El coste del material suma 12,60 euros.

7.2.5 Coste total

Una vez obtenido el desglose de precios se procede a calcular el coste final de proyecto.

Costes parciales	Euros
Coste del personal	9446,13
Coste de hardware	256,81
Coste del software	217,74
Coste de material de oficina	12,50
	9933,28

Tabla 87: Presupuesto coste total sin IVA

A este resultado obtenido se debe sumar un porcentaje del 20% por costes imprevistos y se debe tener en cuenta que ha esto le falta sumar el IVA.

Costes totales	Euros
Coste total sin adiciones	9933,28
20% costes imprevistos	1986,66
	11919,94

Tabla 88: Costes totales

El coste final del proyecto asciende a 11.919,94 euros sin IVA. En el anexo A se puede encontrar el presupuesto firmado para poder firmarlo por el cliente y empezar el proyecto cuando se desee.

8. Conclusiones y mejoras

8.1 Conclusiones

Una vez finalizada la parte de desarrollo y de documentación del proyecto, en este capítulo se hace una recopilación de los objetivos que se plantearon viendo si se han cumplido o no han sido posibles por alguna razón concreta que se explica.

Como podemos recordar el apartado de objetivos lo subdividí en dos: los objetivos profesionales y los personales que tenía que lograr a la conclusión del proyecto. Para poder analizarlos, en este capítulo también voy a dividir para analizar en primer lugar los profesionales y después los personales.

En lo que respecta a la profesional, la palabra que resume adecuadamente todos los objetivos es lograr el proyecto, el algoritmo AHP queda completamente implementado en AngularJS dotado de una aplicación con una experiencia de usuario muy buena que nos permite la aplicación solventar el problema cotidiano de la toma de decisión tanto en la vida cotidiana como empresarial. Todo esto ha sido alcanzado dejando la aplicación como una buena partida de futuros proyectos, ya que ha sido pensada y desarrollada para que posea una escalabilidad y coherencia apta de soportar mejoras que se comentarán a continuación en la siguiente sección del capítulo.

En lo personal, el objetivo era aprender, aprender todas y cada una de las tecnologías que se deciden utilizar, además de las librerías que se incluyen para facilitar casos en el proyecto. Objetivo superado con éxito ya que si de algo tengo que estar orgullosa es de lograr una aplicación de este enfoque. Cuando se me propuso este proyecto me resultó muy atractivo y quise lanzarme a implementarla, he tenido momentos más complicados y otros más llevaderos pero el objetivo se ha cumplido con creces por lo que ha quedado una app digna de ser utilizada por profesionales.

Después de analizar los objetivos que se expusieron al principio quiero citar algunos detalles más concretos con los que me he encontrado a lo largo del desarrollo y caben ser destacados.

Una de las principales dificultades con las que me encontré al comienzo fue respecto a la instalación del entorno de desarrollo. En la universidad, no es hasta el último curso en la asignatura de tecnologías informáticas para la web en el que nos enfrentamos al desarrollo desde cero de una app, por lo que esto me resultó al principio complicado. El hecho de decidir por mí sola las tecnologías con la que iba a ser desarrollado era de gran responsabilidad y yo contaba con poca experiencia como para tomar tal decisión. A esto hay que sumarle la configuración una vez decidido el entorno, que se debe tener mucho cuidado con las versiones que sean compatibles las librerías y el lenguaje porque al subir de versión o bajar puede que no soporte lo mismo que soportaba.

Quiero destacar que el diseño de la interfaz y al tratarse de un proyecto con mucha carga de experiencia de usuario en la parte Font y de UX, el trabajar en una empresa como encargada del departamento de UX en los proyectos me ha facilitado y me ha dado experiencia para afrontar un proyecto de estas características. En 6 meses que llevo en la empresa he aprendido los principios básicos de diseño y a diseñar prototipos de aplicaciones en Sketch que me permiten verlo de una forma muy real ya que se trata prácticamente de una interfaz si le añades la interacción con el programa de inVision.

Debo añadir que la decisión de implementarlo con angular fue para estudiar el lenguaje ya que es el que más se está utilizando desde las empresas del lado de Front, pero al principio al tratarse de un lenguaje no estudiado ni utilizado me costó aprender la lógica que seguía y tuve que desarrollar más despacio que si lo hubiera hecho con un lenguaje conocido.

Otros de los problemas que me he encontrado, podría decir que el mayor de los problemas o el que más me atasqué, es la implementación con angular del algoritmo de decisión multicriterio con AHP, ya que está basado en establecer pesos por pares e ir creando matrices de trazabilidad que luego habrá que operar. Una de las cosas más difíciles de la programación son las matrices ya que se tratan de arrays bidimensionales que además en este caso deben de ser dinámicos porque dependen del árbol de decisión que haga el usuario podrán tener más o menos dimensión.

En resumen, ha sido un proyecto que se ha superado y cumplido con los plazos establecidos en la planificación y con los requisitos por parte del cliente.

8.2 Mejoras

En esta sección expongo algunas de las ideas que se me han ocurrido como posibles mejoras del proyecto que podrán ser desarrollados en trabajos futuros una vez finalizado gracias a la escalabilidad.

- Ahora la aplicación está pensada para que el usuario introduzca uno a uno los candidatos que tiene para su decisión, los cuales tiene que meter sus atributos uno a uno e ir guardándolos para luego aplicar el algoritmo multicriterio AHP. Cabe la posibilidad en este caso de mejora de conectar con una base de datos MongoDB, por ejemplo, que almacene los candidatos propios de ese cliente y de esta forma solo se tendría que realizar el árbol por parte del usuario pulsar sobre continuar y obtener las estadísticas.

- Otra mejora sería a nivel de novedades que se pueda instalar el producto Big Data en la conexión entre base de datos y la app y de esta forma en antena3 se conecte con las audiencias en tiempo real de tal forma que se recibe los datos de las personas que están viendo este proyecto se manda a esta aplicación y se calcula cual es la mejor decisión a sacar para que más gente compre.

- Optimizar la aplicación para poder usarse en dispositivos móviles o tablet utilizando alguna de las herramientas de css como establecer clases dependiendo del tamaño de la pantalla, es decir, hacer una aplicación responsive.

- Permitir personalizar la aplicación para cada cliente. Si se saca el css de toda la app a un archivo único de mastersearch_theme.css me facilita de cara a un cliente cambiar el color corporativo y "hacerlo suyo". De esta forma si quieren la aplicación desde el corte inglés pero solo la compran si es con sus colores corporativos bastaría con ir a este archivo modificar colores y estaría lista para su uso.

Glosario

En esta sección se recogen los acrónimos y términos mencionados a lo largo del documento que son propios de la tecnología y que puede que si no estás metido en esta tecnología no tengas conocimiento de dichos términos.

AHP	Proceso analítico jerárquico
CSS	Hoja de estilos de un programa
HTML	Hoja de texto que ejerce como plantilla de la interfaz.
JavaScript	Lenguaje de programación interpretado.
JQuery	Biblioteca multiplataforma de JavaScript
MVC	Modelo vista controlador
API	Aplicación programada de interfaz
UX	Experiencia de usuario
Front	Parte de la informática encargada de la programación de la parte visual para el usuario.
MCDM	Multiple Criteria Decision Making
AJAX	Asynchronous JavaScrpit and XML
JSON	JavaScript Object Notation
MongoDB	Base de datos
Responsive	Hacer que la aplicación se adapte al tamaño de la pantalla en cada momento y por tanto al dispositivo que se esté utilizando.

Referencias

En esta sección se muestran las referencias utilizadas para el desarrollo del proyecto tanto de cada una de las fases como de la memoria.

[1] Desarrollo con el algoritmo de decisión multicriterio: http://oa.upm.es/37290/

[2] Aplicación similar a la pedida, página utilizada para la parte de investigación del mercado actual. https://riuma.uma.es//xmlui/handle/10630/7769

[3] Algoritmo de decisión multicriterio explicado para entenderlo y luego implementarlo.
http://www.academia.edu/11855187/Algoritmo_de_Ferreira_Decisiones_multicriterio

[4] Fases de un proyecto software: http://proyectosguerrilla.com/blog/2013/02/las-cinco-etapas-en-la-ingenieria-del-software/

[5] Historia de la toma de decisiones para saber el estado del mercado actual https://prezi.com/kd3k-zjoeefh/una-breve-historia-de-la-toma-de-decisiones/

[6] Tecnologías utilizadas a lo largo de la historia para la toma de decisiones https://www.gestiopolis.com/conocimiento-y-tecnologias-de-la-informacion-en-la-toma-de-decisiones/

[7] Aplicaciones para la toma de decisiones http://www.enter.co/chips-bits/apps-software/esta-es-la-aplicacion-perfecta-al-momento-de-tomar-decisiones/

[8] App para la toma de decisión hey! http://mprende.co/tecnol%C3%B3gico/hey-un-app-te-ayuda-en-la-toma-de-decisiones

[9] Cuatro aplicaciones que existen para la ayuda de toma de decisiones en el día a día. http://www.audienciaelectronica.net/2015/03/4-aplicaciones-que-te-ayudaran-a-tomar-mejores-decisiones/

[10] Womity app para la toma de decisiones. http://noticias.universia.es/empleo/noticia/2013/12/16/1069932/womity-toma-decisiones-manera-mas-sencilla.html

[11] Como programar con AngularJS https://carlosazaustre.es/blog/empezando-con-angular-js/

[12] Lógica y arquitectura con AngularJS de los componentes y módulos. http://blog.enriqueoriol.com/2017/03/introduccion-angular-modulo-y-componente.html

[13] Libro de como tomar decisiones en el que te exponen ejemplos con el algoritmo. Saaty, Thomas. How to make a decision: the analytic hierarchy process. University of Pittsburgh. 1994

[14] Librería de angular que me proporciona la lógica de un árbol https://github.com/angular-ui-tree/angular-ui-tree

[15] Librería de angular material https://material.angularjs.org/latest/

[16] Librería de Bootstrap para el desarrollo de la interfaz http://getbootstrap.com/

[17] T. L. Saaty, “The Analytic Hierarchy Process,” Education, pp. 1–11, 1980.

[18] Revistas UTP, Algoritmo AHP multicriterio

[19] T. L. Saaty, “The Analytic Hierarchy Process,” Education, pp. 1–11, 1980.

Anexos

En esta sección se adjuntan dos documentos, por un lado se adjunta el documento de presupuesto que debe firmar el cliente si acepta el proyecto y lo firmaré yo para que quede constancia de que hay un acuerdo entre cliente y jefe de proyecto. Por otro lado existe otra sección de aspectos legales en la que se citan algunos aspectos que deben tenerse en cuenta para esta herramienta web.

Presupuesto

PRESUPUESTO DEL PROYECTO

1º Nombre del Autor del Proyecto.
María Santana Aguila

2º Departamento.
Dep. de Informática.

3º Descripción del Proyecto.

Título:	arrollo de una Herramienta Web para la Resolución de Problemas de Decisión Multicriterio con
Duración:	6,5 meses.
Tasas de Costes Indirectos :	20%

4º Desglose del Presupuesto:

Presupuesto del Personal.

NOMBRE	Descripción	Dedicación	Precio/Hora	Saldo
María Santana Aguila	Analista Junior	105	16,97	1.781,85
María Santana Aguila	Programador Junior	578	13,26	7.664,28
			Total	€ 9.446,13

4º Desglose del Presupuesto:

Presupuesto Coste Hardware.

NOMBRE	Descripción	Dedicación	Precio Unidad	Unidad	Saldo
María Santana Aguila	Macbook Air 17"	6	134,9	1	134,90
María Santana Aguila	Windows Surface Pro	6	110	1	110,00
María Santana Aguila	Disco Duro 1TB	6,5	11,91	1	11,91
				Total	€ 256,81

4º Desglose del Presupuesto:

Presupuesto Coste Software.

NOMBRE	Descripción	Precio Unidad	Unidad	Saldo
María Santana Aguila	Visual Studio Code		1	
María Santana Aguila	Nodejs		1	
María Santana Aguila	Bower		1	
María Santana Aguila	Angular Material		1	
María Santana Aguila	Bootstrap		1	
María Santana Aguila	Karma		1	
María Santana Aguila	Microsoft Office	177,75	1	177,75
María Santana Aguila	Sketch	39,99	1	39,99
			Total	€ 217,74

4º Desglose del Presupuesto:

Presupuesto Material de Oficina.

NOMBRE	Descripción	Precio Unidad	Unidad	Saldo
María Santana Aguila	Cuaderno	6,35	1	6,35
María Santana Aguila	Lapiz	0,30	1	0,30
María Santana Aguila	Bolígrafo	0,50	1	0,50
María Santana Aguila	Goma	0,20	1	0,20
María Santana Aguila	Carpeta	5,25	1	5,25
			Total	€ 12,60

4º Desglose del Presupuesto:

Presupuesto Costes Parciales.

NOMBRE	Descripción	Precio Total	Saldo
María Santana Aguila	Coste de Personal	9.446,13	9.446,13
María Santana Aguila	Coste de Hardware	256,81	256,81
María Santana Aguila	Coste de Software	217,74	217,74
María Santana Aguila	Coste de Material de Oficina	12,60	12,60
		Total	€ 9.933,28

4º Desglose del Presupuesto:

COSTES TOTALES.

NOMBRE	Descripción	Precio Total	Saldo
María Santana Aguila	Coste Total sin Adiciones	9.933,28	9.933,28
María Santana Aguila	20% Costes Imprevistos	1.986,66	1.986,66
		Total	€ 11.919,94

El coste total del proyecto asciende a ONCE MIL NOVECIENTOS DIECINUEVE CON NOVENTA Y CUATRO

Firmado el Cliente.

Firmado por María Santana Águila.

Aspectos legales

Forma jurídica

Para iniciar la actividad empresarial existen diferentes opciones legales que van a marcar el tipo de empresa que deseo crear para desarrollar esta herramienta. Para la formación de esta empresa tendré que elegir por una de las opciones disponibles acerca de la constitución de empresas, entre las cuales se encuentran el registro como sociedad mercantil o emprendedor individual. Opto por la opción de sociedad mercantil limitada ya que es la que más se adapta a las necesidades de la actividad a realizar y mejores beneficios aporta.

Una vez tomada la decisión dentro de la sociedad mercantil de optar por la limitada (S.L). En esta decisión me he decantado por esta opción y no por otras por la perspectiva de negocio, la actividad a desarrollar, el capital de aportaciones sociales y el ratio que te aporta como sociedad.

Se trata de una sociedad de tipo capitalista. El capital social está integrado por las aportaciones iguales, acumulables e indivisibles, que no pueden incorporarse a título negociables ni denominarse acciones. Se aplica sobre pequeñas y medianas empresas de capital reducido en el que intervienen pocos socios con un capital de 3000 euros. Se ajusta a la normativa legal según la Ley 2/1995 de 23 de marzo de Sociedades de responsabilidad limitada.

Procedimiento legal

Debido a la abundancia de trámites legales que se deben realizar para dar de alta una sociedad de estas características, se encargarán dichas acciones en una consultora situada cerca de la empresa en la Comunidad de Madrid que se encargue de estos procedimientos. En resumen, los pasos que debe seguir esta consultoría que me han comentado y que adjunto como visión general son los siguientes:

- **Registro de la empresa en el Registro mercantil central:** Se debe registrar vía telefónica el registro como empresa con denominación de sociedad.
- **Consecución de la escritura pública:** se debe firmar ante notario la escritura de la empresa que deberá contener unos documentos obligatorios que son los siguiente:
  - Denominación social de la empresa
  - Objetivos:
    - Actividades que realiza la nueva empresa
    - Fecha cierre del año fiscal de la empresa.
    - Dónde se va a encontrar el negocio.
    - Patrimonio de cada uno de los propietarios.
    - Plan establecido para la estructura que tendrá el negocio.
- **Registro de HACIENDA (Agencia tributaria)**
- **Registro de la empresa en el Registro Mercantil de la provincia** en este caso la Comunidad de Madrid.

De forma simultánea a este procedimiento que he mencionado paso a paso se deben realizar unas acciones para la actividad a desarrollar con el marco legal y fiscal:

- Registro en la Tesorería general de la Seguridad Social
- Solicitud y posterior adquisición de la licencia en el ayuntamiento de la provincia, en este caso la Comunidad de Madrid.
- Registro de datos personales en la Agencia Española de protección de datos, siguiendo la LOPD, concretamente la Ley de Carácter Personal.

Fiscalidad

La formación de una sociedad limitada S.L compromete a sus usuarios a cumplir unas obligaciones que enumero a continuación:

- Capital mínimo de 3000 euros
- Cumplimiento del modelo 600 y sus apéndices y presentación para su certificación por la Consejería de Hacienda.
- Cumplimiento del modelos 036 y 037 para la inscripción de la empresa en la Declaración Censal.
- Adquisición del código de identificación fiscal.
- Cumplimiento del modelo 200 que supone el respectivo registro y pago del impuesto.

Obtengo una gran ventaja al ser menor de 30 años, durante los 18 meses primeros estoy exento de la tasa de Seguridad Social. Al contratar nuevos empleados se debe seguir la Ley 31/1995, de 8 de Noviembre, de prevención de Riesgos Laborales.

Se deben tener en mente el pago del IVA, la declaración de operaciones con terceros y el impuesto de Transmisiones Patrimoniales.

Patentes

Para evitar la copia y perseguir la protección de las ideas y de la marca creada para esta empresa y de esta forma evitar plagios, copias y demás inconvenientes se debe registrar en la Oficina Española de Patentes y Marcas, el coste global de este trámite es de 120 euros.

Protección de datos

Esta herramienta no trabaja con un login pero cuenta con datos de posibilidad de crear la base de datos que guarde todos los datos que utiliza la empresa que se aplicará en estos casos la LOPD para la protección de datos personales.

Resumen en inglés

1. Introduction

Nowadays and always as I said before, the decisions are on us since we wake up. Cause of technology and the days we are it´s weird that nobody make a tool like that to solve this kind of problems with AI using the decision multi criteria algorithm.

There will be more than one web and mobile application for the resolution of this type of problem, but my proposal is to facilitate and encourage the practice of artificial intelligence and its algorithms to solve this problem.

1.1 Current Context

The advantages that technology gives us, have been very useful in our lives. Within the commercial world technology plays an influential role, it´s clear that we are in the "World 2.0" before a new phase of human development next to technology and with a strong influence and tendency to artificial intelligence.

Such is the speed we evolve today that does not give us time to stop thinking how is changing our days. A few years ago it was unthinkable to go by car sharing a trip with

another person you didn't know (Blablacar) or being able to pay with the mobile pone without having to enter a secret code (Twypcash).

In few years, not far away, we won't have to pay with credit cards, all processes will be increasingly automated and there will be more machines and technology than people.

On the other hand, focusing more on the area that I am going to deal with in this document, the decisions are made and are necessary both in small and large companies as well as in each one of us. I speak as if they were always decisions, but I also include in this field every one of the reports that one wishes to study on a concrete characteristic of different cases.

We are going to study what the current market in this sector puts at our disposal to evaluate in which I will differentiate when implementing this and not another application.

We have a developed application in order to make it easy for hikers to create routes based on their motivation, difficulty and physical capacity. The application executes a recommendation system based on network analysis and on search A* algorithm to calculate the routes that match the user's criteria.

Another case of using this algorithm is a project developed in the university. Waiting lists are a problem for most of the countries with a National Health System. This development proposes to analyze the problem of waiting lists from a Multi-criteria Decision perspective.

After a study of the different existing methodologies, it was decided to elaborate a decision model based on the AHP (Analytic Hierarchy Process) method for the management of waiting lists and is now being applied in the Hospitals of the Community of Madrid.

One more use that the application can offer to us is an electrical power generation problema that was developed in 2013 to avoid and reduce complexity and arguments when making decisions.

This project arises with the idea of updating the scope of decisions in the technological world in which we are in these days. Nowadays almost all people have a mobile phone, tablet or computer to access in to this app.

1.2 Project Description

The project is about the implementation of a web tool based on the multi-criteria decision algorithm, nowadays this algorithm, as I mentioned in the current context, is being used by many companies or for certain applications, but why not create a generic application that we can use in any situation?

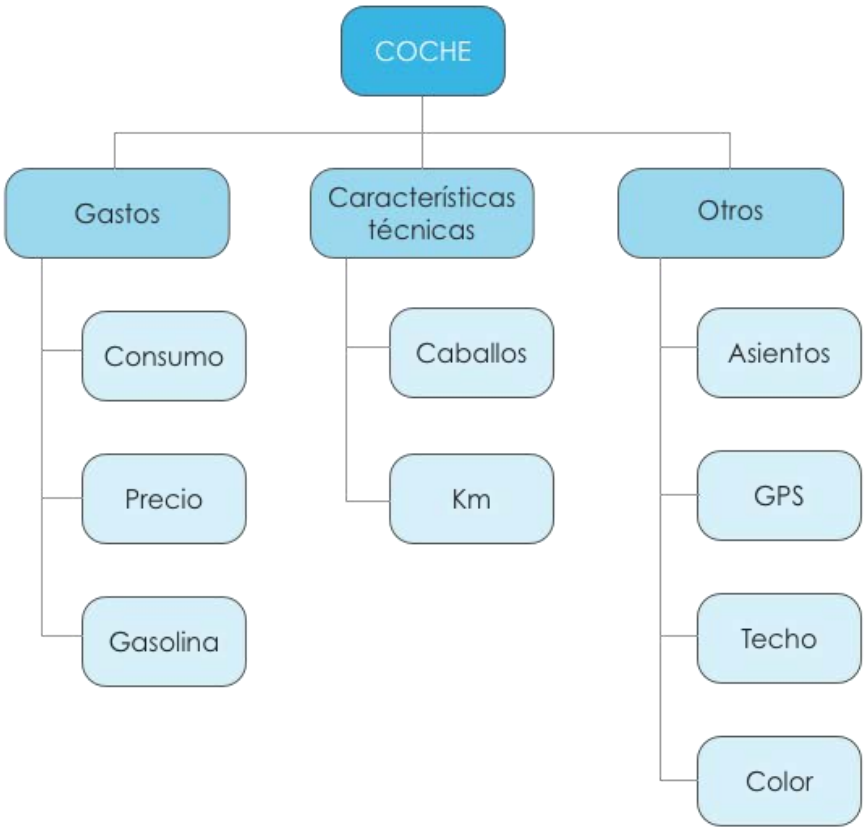
Decision-making requires levels of argumentation based on the information that is available and has been investigated for that decision, as well as the consequences that it'll entail once taken.

Choosing one decision instead of another is the result of a long research process in which it is important to know each other's preferences. Apart from many characteristics that influence the choice, the mood of each of us plays a very important role, if you get up one day with bad mood you can make a different decision than if you are in a good mood. This role is difficult to control but using the multi-criteria algorithm we could put it as one more factor and will influence our choice depending on the relevance we assign.

The multi-criteria decision MCDM (Multiple Criteria Decision Making) is the theory that studies and analyzes the decision problems that involve two or more evaluation criteria. The MCDM is based on framing as accurately as possible, real decision problems that involve different criteria, in which it is very difficult to find the right decision, then it must be the decision maker, applying different techniques, who apply a weight to each of the factors which contemplates and in this way choose the best solution within the set of possible solutions.

The solution of AHP that I've used in the application is based on atomizing the problems and adding the solutions of them. This means, the first step for the application of this method is to structure the problem hierarchically in different interconnected nodes.

Can find a tree example of the levels attached we can set and then a brief description of how that tree would read based on the algorithm and the AHP solution explained in the previous paragraph.



As we can see in the previous graphic as a prototype tree that we would create in the application, the top level is about the purpose of the problem, “the final decision.” The intermediate levels correspond to the different groupings that we make of our factors as we are getting closer to the target we are abstracting some levels. Finally, the levels are the characteristics that I have of each of my candidates for the election, the factors that I will value and I will choose by one decision and not by another.

The AHP application involve:

- Make comparisons in pairs between nodes of each hierarchical level, based on the relevance that the user considers to present to the node of the upper level of the hierarchy to which they are linked. The results of these comparisons are collected as paired comparison matrix.

- Obtain each of the priority vectors corresponding to each of the paired comparison matrix.

- With matrix given in the previous step, the contribution of each alternative to the objective of the problem is calculated through a multiplicative aggregation between the hierarchical levels and according to these results, the decision candidates are sorted and the one that most fits within is selected the established.

- As a last fact, I must add that once given the result that best fits according to the criteria we are valuing, it is possible to perform a sensitivity analysis of the achieved result, visualizing and analyzing each matrix. So you can make the change you want to get another result or following other criteria if they have now changed for various reasons.

Once exposed the algorithm by which this project orbits, I proceed to enter the technical part of its development in great features in addition to first mention the name chosen for the application and why I choose.

The application will be called “Master Search”, because it’s an algorithm search for the best solution according to a series of criteria that each user will establish. Besides that, as an anecdote, when I thought the name that suited the most and in one or two words defined what the tool, when I leave these two match with my initials which finally helped me to decline by this option.

In the research and analysis phase of the current context and cutting edge technologies there are many decisions to make.

- Use Visual Studio Code as a development tool because:
  - Supports all operating systems.

- Clear and simple tool visually and conceptually.
- Easy to deploy to the server the application created using nodejs (bower and npm).
- You can count on the console to add dependencies and install libraries with bower.
- Boot the server with the "gulp serve" statement.

- Bootstrap Library for Angle JS because:
  - Simple and concrete visually.
  - Based on atomic components.
  - It has a lot of good documentation.
  - Coverage of more than 80% in test of each component.

Throughout this document we'll see in more detail each of the implemented screens, the design and the explanation, as well as many other things that are cited in the top index.

Finally, as a summary of this section, it is based on a web tool that allows us to draw the decision tree, thanks to the implementation, with all the mentioned characteristics, of the algorithm according to the established weights will generate a page with the matrix that shows us the solution and we can adjust it or see what has been the determining factor for example.

1.3 Goals

In this section I will divide the objectives into professional and personal to be able to see in Throughout this document we'll see in more detail each of the implemented screens, the design and the explanation, as well as many other things that are cited in the top index.

Finally, as a summary of this section, it is based on a web tool that allows us to draw the decision tree, thanks to the implementation, with all the mentioned characteristics, of the algorithm according to the established weights will generate a page with the matrix that shows us the solution and we can adjust it or see what has been the determining factor for example.

A more differentiated way what they contribute and what I want to achieve in the two areas that both are very important. Once to achieve the scope of the project and the others facing the world of work.

1.3.1 Proffesional Goals

Once the context that influences the application to be developed is known, the objectives to be achieved throughout the project are revealed below. These objectives will help me to draw better conclusions at the end of the document and adjust the degree of closeness between the work performed and the one initially expected.

- Investigate multi-criteria algorithm.
- Create a web application that implements the multi-criteria algorithm.
- Make a prototype that knows all customer requirements.
- Give the application enough design to be very simple and intuitive for the user ("UX": User Expirence).
- Present the layout to the client.
- Describe the development process.
- Implement the application with angularjs.
- Solve the daily problem of business or personal decisions.
- Create a display screen with the results once the candidates have been analyzed.
- Serve as a good match for future projects.

- Create an instruction manual / document in this file that anyone who wants to use the application will see fit.

1.3.2 Personal Goals

Apart from the goals as a project of this work that I mentioned in the previous section, you should always have goals to grow as a person and to train you. In my case, these are my goals at the beginning of the work:

- Learn Angular Language JS.
- Develop an application from scratch alone.
- Learn how to lift the application on the computer.
- Learn how to manage Nodejs.
- Learn the use of npm technology.
- Learn the use and application of bower.
- Learn how to use component design libraries given a language.
- Achieve autonomy and ease with Ajax technology.
- Achieving autonomy and ease with JavaScript technology.
- Investigate and study the multi-criteria decision algorithm.
- Investigate and study the AHP foundation to apply it.
- Investigate competitors' applications.

1.4 Phases of development

At the time of developing a project of these characteristics I followed the typical phases in the development of a software project. It began with a first phase out of the development software research, study and analysis of the current environment, skills, and development of the multi-criteria algorithm.

The whole project had a duration of 6 months and a half or what is the same 141 working days. I detail each of the stages of my project and the duration of each one of them below.

Competition analysis phase

It was the first phase to get an idea of the competition that was in the market and what I was facing. After seeing what I did not find a web application of these characteristics, there were only, as I described in the section of "current context", applications that used this algorithm but for certain things, for example for bicycle route decisions but are for a unique context. This took me 3 days of 7 hours, that is, 21 hours of strict competition analysis.

Problem Investigation Phase

It was the atomically longest phase in the project because of the large number of new technologies and many concepts that I did not know before and had to learn. It had a total duration of one month, that is, 30 calendar days with a total of 7 hours a day, would total a total of 210 hours. Here I include the analysis of the algorithm in such a way to get the logic to get to find the better technology that solves all the problems that I can find in a simple but complete way. After this phase I decided that the technology I was going to use was angularjs that I had not studied in the race but it is very demanded in the market and could serve me face to a future. After deciding this in the last days I had to familiarize myself and take online courses of the tool to understand the syntax in the development, which was a great difficulty to me to be able to understand the logic and connection between components and directives.

Analysis phase

The requirements of the software product to be developed are extracted. At this stage, the experience and ability we have been given at the university to recognize the client's requirements in order to eliminate the incomplete, ambiguous or contradictory are crucial. Usually, the client provides very abstract and ambiguous requirements to what I, as an engineer, have to specify and negotiate with the customer so that he meets their needs to help him obtain a complete vision of the requirements. The main objective of this phase is communication, a very intense stage in which it is necessary to eliminate the maximum ambiguity with it. It had a duration of 49 hours spread over a week (7 hours / 7 days). During this phase I obtained complete documents of use cases and software requirements.

Design and architecture phase

Determines how it will function in general without going into detail the application. It consists of the design of a prototype of the interface that meets the requirements of the previous phase and is usable for the user according to the user experience criteria. Previous diagrams are usually made to know the interactions between the entities and their sequencing. In addition, the system architecture, both physical and logical, is defined. It lasted for 20 days of an average of 6 hours, that is, 120 hours in total.

Phase of development and implementation

Basically, they are summarized in the translation of the previous phase, design, in code. It is the most obvious part of a software project that everyone has with it and is the first to name it, since it is the phase from which "tangible" results are being produced. It does not have to be the longest but it is the phase in which you realize the failures committed in the previous ones, since a failure in a requirement or design can make you go back and re-specify and delay days in this phase of implementation. It lasted 50 days, 6 hours a day, 300 total hours.

Phase of tests

The main objective of this phase is to make sure that the analysis phase is fulfilled, performing a test for each requirement and checking that each negotiated functionality is fulfilled. In it, some tests are included in the test plan, and they are carried out. It took 15 days, 5 of each test specification and 10 implementation, with a total of 75 hours.

Phase of documentation

This phase is carried out in parallel to all the previous ones and it is in which this document is realized, that is to say, as objective it has to obtain a memory of the complete project that answers any doubt that someone who wishes know the project. It had two parts; the first, which was developed at the same time as I said not to forget what I was doing each day was the previous of each mentioned phase plus 2 hours of documentation, and the second, was at the end of the test phase in which the application is given as completed to fill in the sections that should be done later as conclusions, phase hours, or other of these characteristics. It had a total duration of 141 days that lasted the project distributed in the two phases described.

2. Conclusion and improvements

Once the development and documentation part of the project is completed, this chapter compiles the goals that were considered if they have been fulfilled or have not been possible for some specific reason that is explained.

As we can remember the section of goals subdivided it into two: the professional and personal goals that had to achieve at the conclusion of the project. In order to analyze them, in this chapter I will also divide to analyze first the professionals and then the personal ones.

With respect to the professional, the word that adequately summarizes all the goals is to achieve the project, the algorithm AHP is completely implemented in AngularJS endowed with an application with a very good user experience that allows the application to solve the daily problem of decision making in everyday life as well as business. All this has been achieved leaving the application as a good starting point for future projects, as it has been designed and developed to have a scalability and coherence apt to support improvements that will be discussed in the next section of the chapter below.

Personally, the goal was to learn, learn each and every one of the technologies that are decided to use, in addition to the libraries that are included to facilitate cases in the project. A goal successfully passed since if anything I have to be proud is to achieve an application of this approach. When I was proposed this project was very attractive and I wanted to launch myself to implement it, I have had more complicated times and others more bearable but the objective has been fulfilled by what has remained an app worthy of being used by professionals.

After analyzing the objectives that were set out at the beginning I want to cite some more specific details with which I have found throughout the development and can be highlighted.

One of the main difficulties I encountered at the beginning was with the installation of the development environment. At university, it is not until the last course in the subject of computer technologies for the web that we are faced with the development from scratch of an app, so this was complicated at first. Deciding on my own the technologies with which I was to be developed was of great responsibility and I had little experience to make such a decision. To this we must add the configuration once the environment is decided, we must be very careful with the versions that are compatible libraries and language because when uploading version or download may not support the same as supported.

I want to point out that the design of the interface and being a project with a lot of user experience in the Font and UX part, working in a company as a UX department manager in the projects has facilitated and given me experience to deal with such a project. In 6 months I have been in the company I learned the basic principles of design and design prototypes of applications in Sketch that allow me to see it in a very real way since it is practically an interface if you add the interaction with the program of inVision.

I must add that the decision to implement it with angular was to study the language since it is the one that is being used the most from the side of Front, but at first being a language not studied or used I struggled to learn the logic that followed and I had to develop more slowly than if I had done so with a familiar language.

Another of the problems that I have found, I could say that the biggest problem or the one that I got stuck in, is the implementation with angular of the multicriteria decision algorithm with AHP, since it is based on establishing weights by pairs and creating arrays of traceability that will then have to operate. One of the most difficult things of programming are the arrays because they are two-dimensional arrays that in this case must be dynamic because they depend on the decision tree that the user can have more or less dimension.

In short, it has been a project that has been exceeded and met the deadlines set in the planning and with the requirements by the client.

2.2 Improvements

In this section I outline some of the ideas that have occurred to me as possible project improvements that can be developed in future works once finished thanks to the scalability.

- ✓ Now the application is designed for the user to enter one by one the candidates that have for their decision, which has to put their attributes one by one and save them to then apply the algorithm multi-criteria AHP. It is possible in this case to improve the connection with a MongoDB database, for example, to store the candidates of that client and thus only have to perform the tree on the part of the user click on continue and obtain statistics.

- ✓ Another improvement would be in terms of news that the Big Data product can be installed in the connection between the database and the app and thus in antenna3 connect with the audiences in real time in

such a way that data is received from the people who are watching this project is sent to this application and it is calculated which is the best decision to make so that more people buy.

- ✓ Optimize the application to be able to be used in mobile devices or tablet using some of the tools of css like to establish classes depending on the size of the screen, that is to say, to make a responsive application.

- ✓ Allow customization of the application for each client. Taking the css from the entire app to a single mastersearch_theme.css file makes it easy for a customer to change corporate color and "make it yours". This way if you want the application from the English cut but only buy it if it is with your corporate colors just go to this file to modify colors and it would be ready for use.